intel®

E^2 Prom Family Applications Handbook

Book II

Order No. 210273-001

intel

E^2PROM FAMILY APPLICATIONS HANDBOOK

BOOK II

NOVEMBER 1981

Intel Corporation makes no warranty for the use of its products and assumes no responsibility for any errors which may appear in this document nor does it make a commitment to update the information contained herein.

The following are trademarks of Intel Corporation and may only be used to identify Intel Products:

BXP, CREDIT, i, ICE, iCS, i_m, Insite, Intel, int$_e$l, Intelevision, Intellec, iRMX, iSBC, iSBX, Library Manager, MCS, Megachassis, Micro-mainframe, Micromap, Multimodule, Plug-A-Bubble, PROMPT, RMX/80, System 2000 and UPI.

MDS is an ordering code only and is not used as a product name or trademark. MDS® is a registered trademark of Mohawk Data Sciences Corporation.

*MULTIBUS is a patented Intel bus.

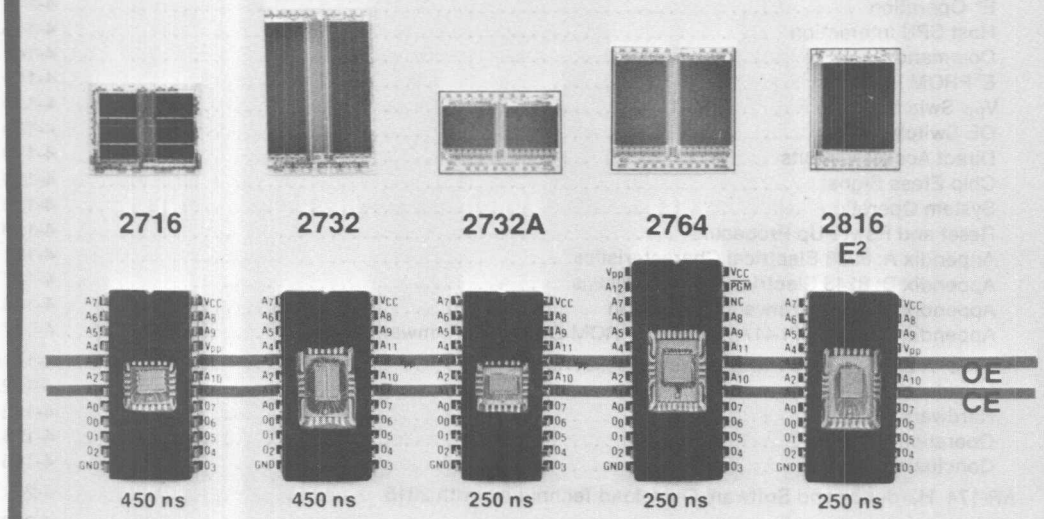
Additional copies of this manual or other Intel literature may be obtained from:

Intel Corporation
Literature Department SV3-3
3065 Bowers Avenue
Santa Clara, CA 95051

© INTEL CORPORATION, 1981

Table of Contents

Intel's EPROM and E² Family



Introduction 1

Introduction

During the past ten years, Intel has developed EPROMs to meet the needs of the most demanding customer systems. The quest for a perfect non-volatile memory has been led by Intel from ROM to PROM to EPROM and now, after intense development, to the E^2PROM. The E^2PROM technology promises to alter dramatically the microprocessor systems of today and offer end users greatly enhanced flexibility and system cost-effectiveness.

With regard to adding functions and benefits to your systems, only you can understand the doors that the 2816 will open. Intel is committed to the technology of electrically erasable PROMs and we see it as truly a revolution in non-volatile memory.

Within this handbook are articles, application notes, application briefs, and other data which will tell you all you need to know to design the E^2PROM into your system today.

If you would like further information, contact one of the Intel Sales Representatives listed in the back or return the reply card.

ACKNOWLEDGEMENT

The author wishes to acknowledge the following contributors to the 2816 development and introduction: K. Armstrong, A. Baluni, B.L. Barfield, N. Boruta, R. Battat, A. Chan, V. Dham, B. Euzent, H. Fung, G. Gongwer, K. Gudger, W.S. Johnson, L.N. Jordan, D. Kijanka, P. Mareno, D. Oto, B. Pochowski, D. Schaedler, B. Shiner, and R. Wood.

Without their limitless perseverance and dedication, the 2816 would not have been possible.

E²PROM Backgrounder Information

2

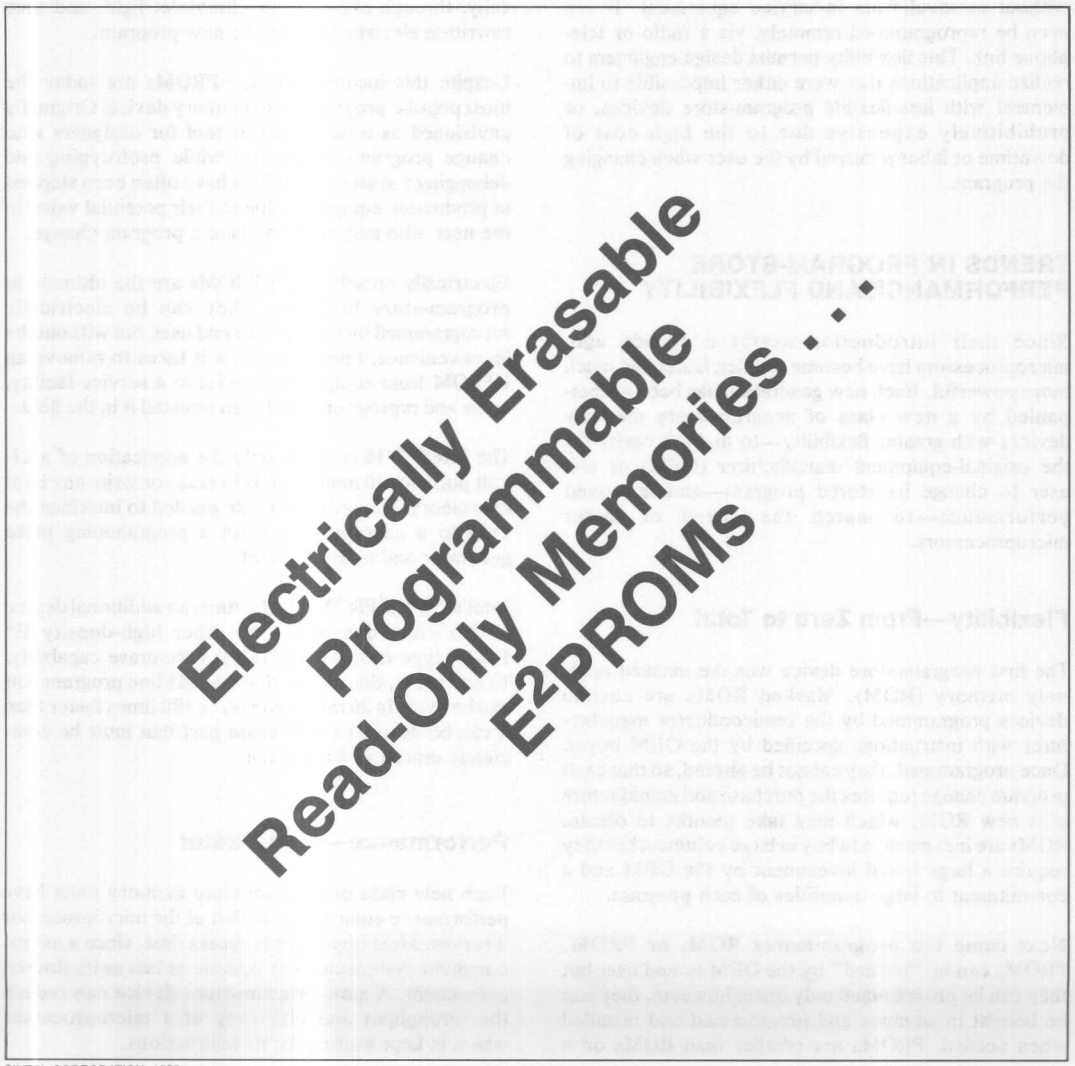
EPROM Background Information

2

intel

E²PROM BACKGROUNDER

November 1980



©INTEL CORPORATION, 1980.

INTRODUCTION

Intel Corporation, the leading manufacturer of microprocessors, semiconductor memories and microcomputer system components, has just introduced its first electrically erasable programmable read-only memory (E^2PROM). Designated the 2816, this 16-kilobit E^2PROM is the first of a new breed of memory that will eventually become the standard storage medium for microprocessor programs.

Not only is the 2816 non-volatile, fully static and fast enough to support a high-performance microprocessor, but it can be reprogrammed electrically in the field, without removal from in-service equipment. It can even be reprogrammed remotely, via a radio or telephone link. This flexibility permits design engineers to realize applications that were either impossible to implement with less-flexible program-store devices, or prohibitively expensive due to the high cost of downtime or labor incurred by the user when changing the program.

TRENDS IN PROGRAM-STORE PERFORMANCE AND FLEXIBILITY

Since their introduction nearly a decade ago, microprocessors have become smaller, faster and much more powerful. Each new generation has been accompanied by a new class of program-store memory devices with greater flexibility—to make it easier for the original-equipment manufacturer (OEM) or end user to change its stored program—and improved performance—to match the speed of faster microprocessors.

Flexibility—From Zero to Total

The first program-store device was the masked read-only memory (ROM). Masked ROMs are custom devices programmed by the semiconductor manufacturer with instructions specified by the OEM buyer. Once programmed, they cannot be altered, so that each program change requires the purchase and manufacture of a new ROM, which may take months to obtain. ROMs are inexpensive to buy in large volumes, but they require a large initial investment by the OEM and a commitment to large quantities of each program.

Next came the programmable ROM, or PROM. PROMs can be "burned" by the OEM or end user but they can be programmed only once; however, they can be bought in advance and programmed and installed when needed. PROMs are costlier than ROMs on a per-unit basis, but they eliminate the risk and wait for delivery of a new batch of masked ROMs from the semiconductor manufacturer.

Erasable PROMs, or EPROMs, added considerable flexibility to the programming step. Like PROMs, EPROMs can be stocked and programmed by the OEM or end user, but they can be reprogrammed thousands of times. This eliminates the need to scrap expensive parts each time a program change is needed.

With regard to flexibility, the only drawback to EPROMs is that they must be removed from the equipment to be reprogrammed. EPROMs are erased optically, through exposure to ultraviolet light, and then rewritten electrically with the new program.

Despite this inconvenience, EPROMs are today the most popular program-store memory device. Originally envisioned as a development tool for designers who change programs frequently while prototyping and debugging a system, EPROMs have often been shipped in production equipment due to their potential value to the user who may wish to make a program change.

Electrically erasable (E^2) PROMs are the ultimate in program-store flexibility. They can be electrically reprogrammed by the OEM or end user, but without the inconvenience, time or expense it takes to remove an EPROM from equipment, send it to a service facility, erase and reprogram it and then reinstall it in the field.

The Intel® 2816 requires only the application of a 21-volt pulse for 10 milliseconds to erase or write any byte of memory. The only hardware needed to interface the 2816 to a microprocessor are a programming pulse generator and a timer circuit.

Intel's 2816 E^2PROM also features an additional degree of flexibility unmatched by other high-density E^2 PROM-type devices: individual byte-erase capability. To end users, this means that a single line program edit can be made in 20 milliseconds, or 100 times faster than it can be done on a bulk-erase part that must be completely erased and rewritten.

Performance—Ever Faster

Each new class of program-store memory must have performance comparable to that of the microprocessor it serves. Most important is access time, since a microcomputer system can only operate as fast as its slowest component. A slow program-store device can reduce the throughput and efficiency of a microprocessor which is kept waiting for its instructions.

A recent trend which affects program-store memories is toward more complex systems, with multiplexed address and data lines. Program-store memories must be able to be precisely controlled by the microprocessor, to ensure that they do not read instructions onto the bus when the microprocessor is not expecting them.

The 2816 E²PROM has both the speed and controllability required for service in a state-of-the-art microcomputer system. It has an access time of 250 nanoseconds, which is fast enough to eliminate the need to insert so-called wait states in a high-performance microprocessor's program, just to allow for slow program memory.

The 2816 also features Two-line control, a system-control function that has become essential in large, high-speed microcomputer systems. Two-line control eliminates contentions between addresses and data on bus lines. The chip has separate output-enable and chip-enable pins that permit the microprocessor to control exactly when it is enabled.

In addition, the 2816 comes in a 24-pin package that conforms to the new industry-standard pinout for high-density, byte-wide memories recently approved by the Joint Electron Device Engineering Council (JEDEC). By using the 2816 and printed-circuit boards with 28-pin sockets, system designers can be assured of future compatibility and interchangeability of microcomputer-system memory components up to 256 kilobits in density.

IMPLICATIONS AND APPLICATIONS

E²PROMs will have a profound impact on microcomputer system design. As designers learn to fully use their flexibility, E²PROMs' cost per function will fall dramatically through greater design efficiency.

The semiconductor cost/volume learning curve will reduce E²PROM prices to parity with EPROMs by the mid-1980s, when they will replace EPROMs as the standard program-store medium in microprocessor-based equipment. In the interim, E²PROMs will be designed into those applications where their cost is offset by the functional value their flexibility adds to the end-user product.

Near-Term Applications

One market segment that will find E²PROMs attractive immediately is industrial process control. In large plants with distributed processing stations under control of a central computer, E²PROMs can improve local process monitoring and control.

In such configurations, the central computer alters the E²PROMs' contents remotely when a change in process occurs, to optimize local processor operation to the new conditions. The E²PROMs can also be used as data store devices to monitor flow rates, value closures and like information, freeing the central computer for more important duties.

Another obvious application for E²PROMs today is as replacements for core memory or fuse-link PROMs in military equipment and commercial aircraft. Here, the cost of an E²PROM is more than offset by the alternative cost of replacing expensive parts each time the user wishes to change flight coordinates or radio frequencies.

Point-of-sale (POS) terminals are an ideal application for E²PROMs, where they function as look-up tables whose contents—product pricing, for example—do not change frequently. The central computer can poll and update the E²PROMs after business hours of the retail store, to monitor sales volumes and adjust pricing to inflation.

Another application for E²PROMs is in programmable robots like those used in automobile manufacturing or industrial metalworking. Presently, program changes require replacing the paper or magnetic tape that controls the robot's operation. An alterable, non-volatile semiconductor memory like the 2816 has distinct advantages here, especially in light of its superior reliability in dirty industrial environments. Besides its ability to be reprogrammed quickly and remotely by a central computer, an E²PROM can easily pay for itself by avoiding retooling charges and by preventing failures that could destroy an expensive piece of material.

February 1980

A 16K Electrically Erasable Nonvolatile Memory

Presented at the IEEE
International Solid State
Circuitry Conference.
February 1980

SESSION XII: ROMs, PROMs AND EROMs

THPM 12.6: A 16Kb Electrically Erasable Nonvolatile Memory

William S. Johnson, George Perlegos, Alan Renninger, Greg Kuhn and T. R. Ranganath[†]

Intel Corp.

Santa Clara, CA

FLOATING GATE STRUCTURES have been highly successful as nonvolatile devices because of their compatibility with silicon gate processing and their excellent charge retentivity with applied voltage at operating temperature. The accepted method of erasure in the commercial marketplace is ultra-violet light (EPROM)[1], although proposals have been made to erase electrically by avalanche injection of holes[2], electron tunneling[3,4], or a combination of both[5]. These methods, however, have typically suffered from poor reproducibility and very fast wearout during program/erase cycling.

To realize nonvolatile devices which can be erased electrically with high program/erase endurance, many have resorted to MNOS structures[6] which are programmed and erased by direct tunneling through a thin oxide. In this approach, charge is stored in traps within the nitride dielectric. A major problem with this approach is that the properties of the nitride/oxide dielectric are difficult to control and are adversely affected by normal silicon gate processing. Furthermore, the threshold voltages of these structures are vulnerable to disturbance by even small applied voltages and data retention is not easily guaranteed for long periods (years).

The device reported (FLOTOX, for *f*loating gate *t*unnel *o*xide) retains the processing and the retention advantages of floating gate over MNOS while solving the traditional endurance problem. This is accomplished by utilizing an oxide less than 200Å thick between a floating poly gate and an N^+ region, as shown in Figure 1. In FLOTOX both program and erase are accomplished by tunneling[7] of electrons through the tunnel oxide using voltages of less than 25V. A typical endurance plot for a single cell appears in Figure 2. This shows that the threshold window remains open beyond 100,000 cycles. Also by keeping voltages low during read, this structure can retain charge over 10 years under full power, at operating temperatures. There is no refresh requirement no matter how many read accesses are made.

The FLOTOX cell configuration, shown in Figure 3, uses two devices, a select transistor and a memory transistor. Cell area is $0.85mil^2$. Clearing of the memory is accomplished by programming every device in a row. This is done by selecting a row and raising the program line to VPP, which attracts electrons to the floating gate. Writing is accomplished by erasing selected bits within a word. This is done by again selecting a row, but now the program line is held at zero volts while selected columns go to VPP. Electrons are thus removed from the floating gates of the selected devices.

Figure 4 shows the 16K chip, which is arranged as 2K/8b words. It is packaged with 24 leads with a pinout identical to the 16K EPROM*. The chip is automatically powered down until selected ($\overline{CE}$ low). Read is accomplished by selecting the part and enabling the output buffers ($\overline{OE}$ low). On the other hand, selecting the part and taking VPP to 20V for 10ms puts the chip in write mode and writes a word. If the incoming data are all 1's, then the chip automatically goes into clear mode and clears the addressed word. Thus, a clear-write sequence requires merely two 10ms writes, first all 1s, then the data desired. If clearing of the entire chip is desired, this can be accomplished with one 10ms pulse by applying VPP to $\overline{OE}$ as well as the VPP pin with the chip selected. This approach allows a wide variety of functions while maintaining simple control and complete EPROM compatibility.

FLOTOX utilizes a new high performance N-channel two-level-poly silicon gate technology with channel lengths of 3.5μ. Access times for the 16K FLOTOX E^2PROM are below 200ns as shown in Figure 5. This allows use of the device with the newer microprocessors which operate in the 5-8MHz range without wait states. Other features of the 16K E^2PROM are listed in the table.

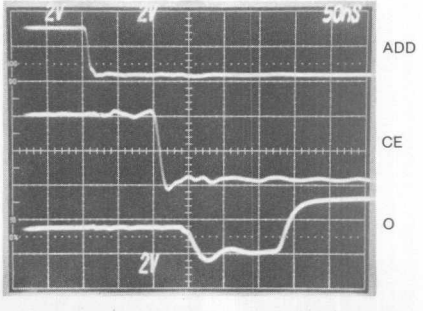


FIGURE 5—Access time for E^2PROM.

[†]Current Address: Hughes Research, Malibu, CA

*2716.

[1]Salsbury, P.J., Morgan, W.L., Perlegos, G. and Simko, R.T., "High Performance MOS EPROMs Using A Stacked Gate Cell", *ISSCC DIGEST OF TECHNICAL PAPERS*, p. 186; Feb., 1977.

[2]Gosney, W.M., "DIFMOS — A Floating-Gate Electrically Erasable Nonvolatile Semiconductor Memory Technology", *IEEE Transactions on Electron Devices*, ED-24, p. 594; May, 1977.

[3]Gulterman, D.C., Rimari, I.H., Halvorson, R.D., McElroy, D.J. and Chan, W.W., "Electrically Alterable Hot-Electron Injection Floating Gate MOS Memory Cell With Series Enhancement", *IEDM Technical Digest*, p. 340; Dec., 1978.

[4]Harari, E., Schmitz, L., Troutman, B. and Wang, S., "A 256-Bit Nonvolatile Static RAM", *ISSCC DIGEST OF TECHNICAL PAPERS*, p. 108; Feb., 1978.

[5]Scheibe, A. and Schulte, H., "Technology of a New N-Channel One-Transistor EAROM Cell Called SIMOS", *IEEE Transactions on Electron Devices*, ED-24, p. 600; May, 1977.

[6]Hagiwara, T., Kondo, R., Yatusuda, Y., Minami, S. and Itoh, Y., "A 16Kb Electrically Erasable Programmable ROM", *ISSCC DIGEST OF TECHNICAL PAPERS*, p. 50; Feb., 1979.

[7]Lenzlinger, M. and Snow, E.H., "Fowler-Nordheim Tunnelling into Thermally Grown SiO_2", *J. of Applied Physics*, 40, p. 278-283; Jan., 1969.

©1980 Institute of Electrical and Electronic Engineers, all rights reserved.

	16K E^2PROM	*16K EPROM*
Configuration	2K X 8	2K X 8
Package	24 pin	24 pin
Power Supplies		
read mode	+5	+5
clear/write	+5, +20	+5, +25
Write		
method	tunnel injection	hot electron injection
time/word	10ms	50ms
Clear		
method	tunnel ejection	UV light
time/word	10ms	–
time/chip	10ms	30 min
Access Time	200ns	450ns
Power Dissipation		
active	500mW	550mW
standby	100mW	100mW
Data Retention	10 years	10 years
Refresh Requirement	None	None

TABLE 1

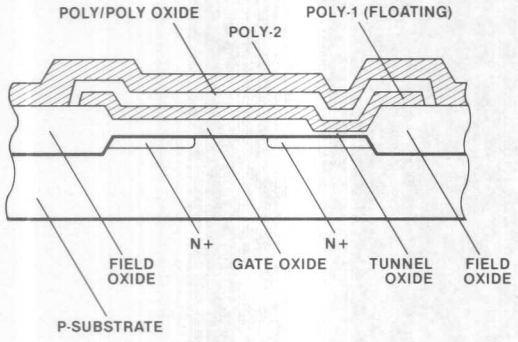


FIGURE 1–Cross section of memory transistor.

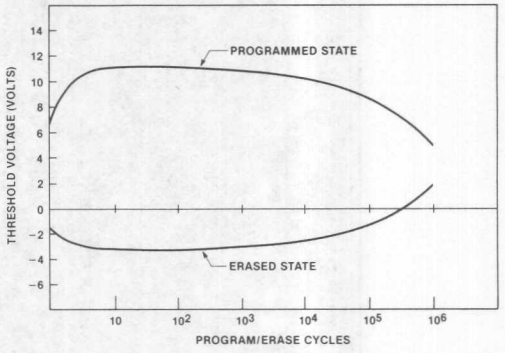


FIGURE 2–Program/erase endurance for single cell.

[See page 271 for Figure 4.]

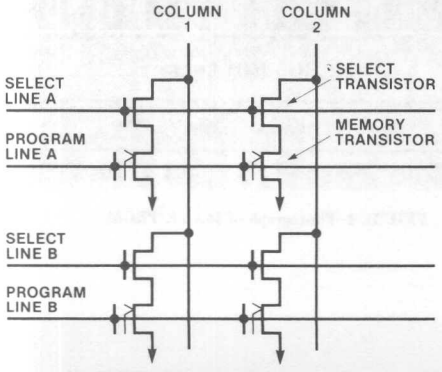


FIGURE 3–Schematic of memory cells.

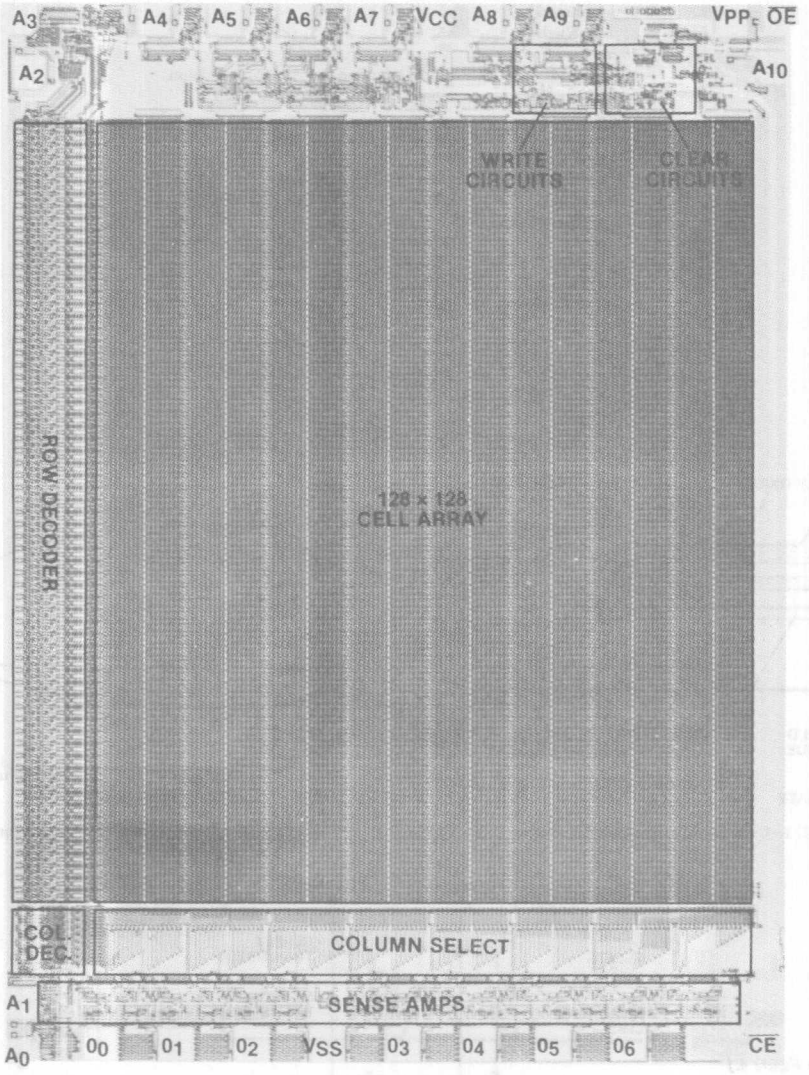


FIGURE 4—Photograph of 16Kb E^2PROM.

intel®

ARTICLE REPRINT

AR-119

March 1980

16-K EE-PROM Relies On Tunneling For Byte-Erasable Program Storage

W. S. Johnson, G. L. Kuhn, A. L. Renninger, and G. Perlegos

Electronics, February 28, 1980

Reprinted from *ELECTRONICS*, February 28, 1980; Copyright © McGraw-Hill, Inc., 1980. All rights reserved.

The electrically erasable programmable read-only memory, or EE-PROM, will one day be the standard form of program storage in microprocessor-based systems. It will follow in the steps of the ultraviolet-light–erasable PROM, for it, too, will become available in increasingly larger byte-wide arrays and will in time share silicon with single-chip microcomputers.

As with the E-PROM, the success of the EE-PROM described in this article hinges upon the mastery of a difficult process. The floating-gate avalanche cell, also pioneered by Intel, is a tricky construction that still eludes many a memory maker. Likewise, the widespread availability of large EE-PROMs is still years off.

The EE-PROM process will be perfected, though, because the rewards go beyond the elimination of the expensive quartz window on the E-PROM package. The electrically erasable memory will usher in systems previously not practical. The microprocessor system whose programs can be altered remotely, as by phone, is one example. Another is the system that is immune to power outages, as it protects its contents in ROM. Perhaps most important, systems will be able to adjust their own program memory to environmental changes.

To be sure, there is more than one way to build an EE-PROM. The metal-nitride-oxide-semiconductor (MNOS) structure has served for years in modest-sized arrays for TV tuning applications, for example. In fact, a year ago Hitachi Ltd. announced a 2-K-by-8-bit MNOS replacement for the 2716 E-PROM. Compatibility with the 2716 is the impetus behind the device described in the following article, but it uses only silicon and its derivatives, plus metal. Also, in place of avalanche injection, which can injure a cell, electrons tunnel to and from a floating gate. **-John G. Posa**

16-K EE-PROM relies on tunneling for byte-erasable program storage

Thin oxide is key to floating-gate tunnel-oxide (Flotox) process used in 2,048-by-8-bit replacement for UV-light–erasable 2716 E-PROM

by W. S. Johnson, G. L. Kuhn, A. L. Renninger, and G. Perlegos, *Intel Corp., Santa Clara, Calif.*

☐ The erasable programmable read-only memory, or E-PROM, is the workhorse program memory for microprocessor-based systems. It is able to retain data for years, and it can be reprogrammed, but to clear out its contents for new data, ultraviolet light must be made to stream through its quartz window. This works well for many applications, but the technique foregoes single-byte—in favor of bulk—erasure and in-circuit self-modification schemes.

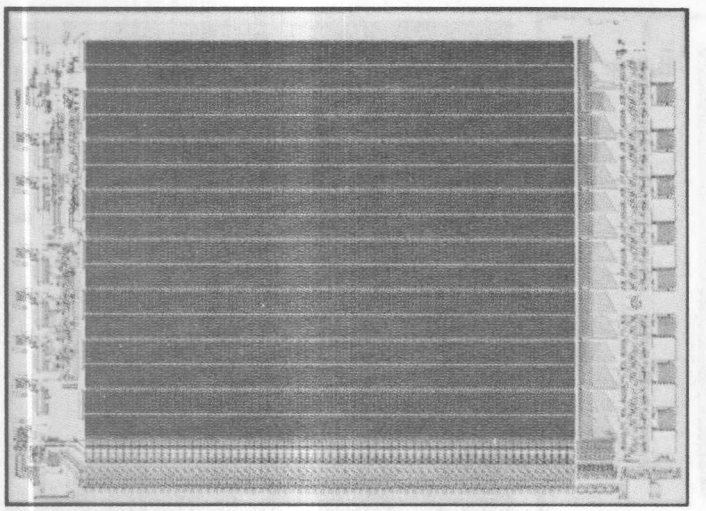
Electrical erasability is clearly the next step for such memories, but like ultraviolet erasure a few years back, it is hard to achieve. In fact, the design of an electrically erasable read-only memory is paradoxical. In each cell, charge must somehow be injected into a storage node in a matter of milliseconds. Once trapped, however, this charge may have to stay put for years while still allowing the cell to be read millions of times. Although these criteria are easily met individually, the combination makes for a design with conflicting requirements.

These demands are more than met in a new EE-PROM, which is a fully static, 2-K-by-8-bit, byte- or chip-erasable nonvolatile memory. At 16,384 bits, this new design not only meets the goal of high density, but also has long-term retention, high performance, and no refreshing requirement, in addition to functional simplicity unmatched by present nonvolatile memories. The device need not be removed from a board for alterations, and performance is consistent with the latest generation of 16-bit microprocessors such as the 8086.

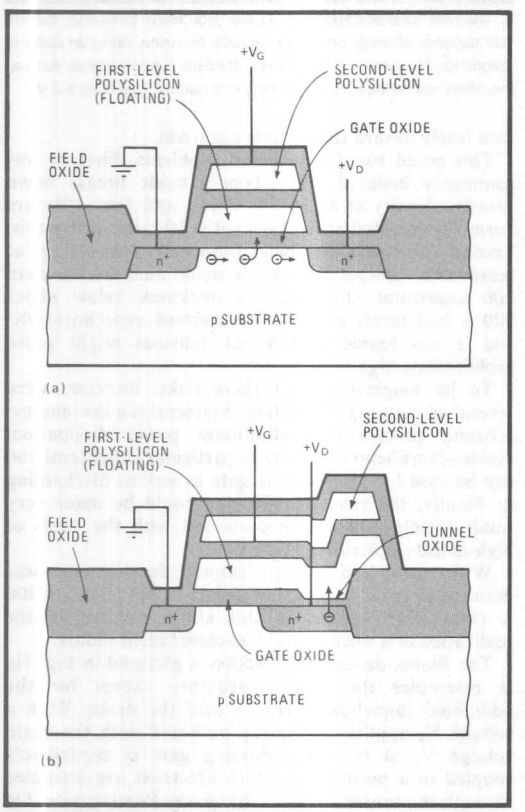
This achievement required the development of a new nonvolatile process technology, HMOS-E, as well as a new cell structure, Flotox, for floating-gate tunnel oxide.

Conflicting requirements

Nonvolatile semiconductor memories generally store information in the form of electron charge. At cell sizes achievable today, this charge is represented by a few million electrons. To store that many electrons in a 10-millisecond program cycle requires an average current on the order of 10^{-10} amperes. On the other hand, if it is essential that less than 10% of this charge leaks away in 10 years, then a leakage current on the order of



The next memory. The 16-K electrically erasable programmable read-only memory is eminently suitable for microprocessor program storage. Organized as 2,048 by 8 bits, the EE-PROM allows full-chip or individual-byte erasure using the same supply (V_{DD}) as for programming.



1. First Famos, now Flotox. The Famos cell (a) found in all E-PROMs stores charge on the floating gate by avalanche means. Flotox cell (b), the heart of the EE-PROM, relies on electron tunneling through thin oxide to charge and discharge the floating gate.

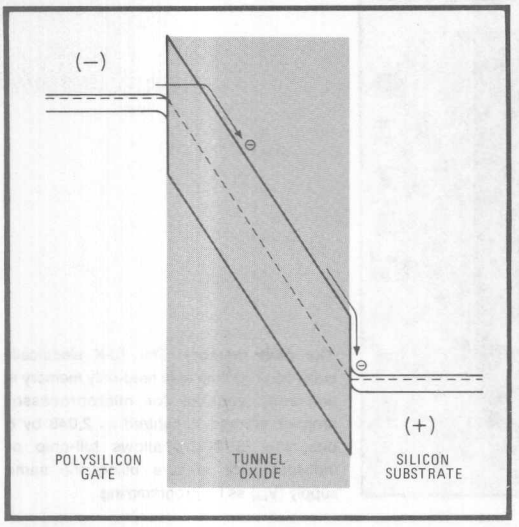
10^{-21} A or less must be guaranteed during read or storage operations. The ratio of these currents, $1:10^{11}$, represents a difficult design problem. Few charge-injecting mechanisms are known that can be turned off reliably during nonprogram periods for such a ratio.

One structure that has proven capable of meeting such stringent reliability requirements has done so for many millions of devices over the last nine years. This is the floating-gate avalanche-injection MOS (Famos) device used in the 1702, 2708, 2716, and 2732 E-PROM families. In the Famos structure, shown in Fig. 1a, a polysilicon gate is completely surrounded by silicon dioxide, one of the best insulators around. This ensures the low leakage and long-term data retention.

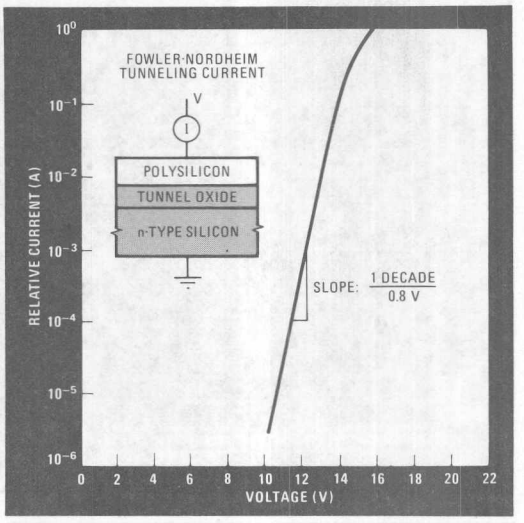
To charge the floating gate, electrons in the underlying MOS device are excited by high electric fields in the channel, enabling them to jump the silicon/silicon-dioxide energy barrier between the substrate and the thin gate dielectric. Once they penetrate the gate oxide, the electrons flow easily toward the floating gate as it was previously capacitively coupled with a positive bias to attract them.

Because of Famos' proven reliability, the floating-gate approach was favored for the EE-PROM. The problem, of course, was to find a way to discharge the floating gate electrically. In an E-PROM, this discharge is effected by exposing the device to ultraviolet light. Electrons absorb photons from the UV radiation and gain enough energy to jump the silicon/silicon-dioxide energy barrier in the reverse direction as they return to the substrate. This suffices for off-board program rewriting, but the object of the EE-PROM is to satisfy new applications that demand numerous alterations of the stored data without removing the memory from its system environment. What evolved was the new cell structure called Flotox (Fig. 1b).

In the quest for electrical erasability, many methods were considered, and several potentially viable solutions were pursued experimentally. One initially attractive



2. Tunneling. For a thin enough oxide, as shown here, under a field strength of 10^7 V/cm, Fowler-Nordheim tunneling predicts that a certain number of electrons will acquire enough energy to jump the forbidden gap and make it from the gate to the substrate.



3. Current characteristic. In Fowler-Nordheim tunneling, current flow depends strongly on voltage across the oxide, rising an order of magnitude for every 0.8 V. Charge retention is adequate so long as the difference between programming and reading is at least 8.8 V.

approach attempts to harness a parasitic charge-loss mechanism discovered in the earliest E-PROMs. Referring again to Fig. 1a, the polysilicon grains on the top surface of the floating gate tend, under certain processing conditions, to form sharp points called asperities. The sharpness of the asperities creates a very high local electric field between the polysilicon layers, shoving electrons from the floating gate toward the second level of polysilicon. This effect is purposely subdued in today's E-PROMs by controlling oxide growth on top of the floating gate because this parasitic electron-injection mechanism would otherwise interfere with proper E-PROM programming.

It was first thought that asperity injection could be used to erase the chip. In fact, fully functional, electrically erasable test devices were produced; but the phenomenon proved unreproducible and the devices tended to wear out quickly after repeated program and erase cycling. After over a year's effort, that approach was abandoned.

Tunneling solution

The solution turned out to be the one that initially seemed impossible. After investigating many methods of producing energetic electrons, it was decided to approach the problem from a different direction: to pass low-energy electrons through the oxide. This could be accomplished through Fowler-Nordheim tunneling, a well-known mechanism, depicted by the band diagram in Fig. 2. Basically, when the electric field applied across an insulator exceeds approximately 10^7 volts per centimeter, electrons from the negative electrode (the polysilicon in Fig. 2) can pass a short distance through the forbidden gap of the insulator and enter the conduction band. Upon their arrival there, the electrons flow freely toward the positive electrode.

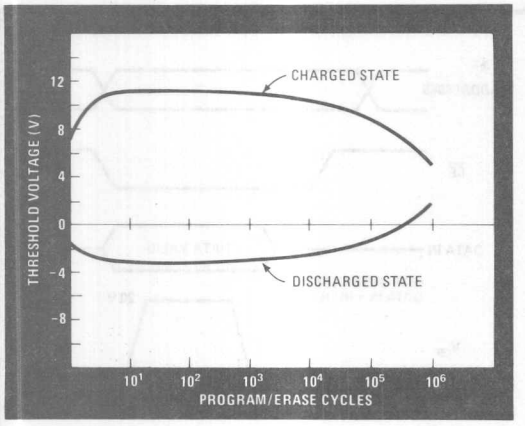
This posed two fundamental problems. First, it was commonly believed that silicon dioxide breaks down catastrophically at about 10^7 V/cm, and MOS FETs are normally operated at field strengths 10 times below this. Second, to induce Fowler-Nordheim tunneling at reasonable voltages (20 V), the oxide must be less than 200 angstroms thick. Oxide thickness below about 500 Å had rarely even been attempted experimentally, and it was feared that defect densities might prove prohibitively high.

To be weighed against these risks, however, were several advantages. Tunneling in general is a low-energy, efficient process that eliminates power dissipation. Fowler-Nordheim tunneling in particular is bilateral and can be used for charging the gate as well as discharging it. Finally, the tunnel oxide area could be made very small, which is of course consistent with the needs of high-density processing.

With these motivating factors, development was initiated to grow reliable, low-defect oxides less than 200 Å thick. The success of this effort resulted in the realization of a working cell structure called Flotox.

The Flotox device cross section is pictured in Fig. 1b. It resembles the Famos structure except for the additional tunnel-oxide region over the drain. With a voltage V_g applied to the top gate and with the drain voltage V_d at 0 V, the floating gate is capacitively coupled to a positive potential. Electrons are attracted through the tunnel oxide to charge the floating gate. On the other hand, applying a positive potential to the drain and grounding the gate reverses the process to discharge the floating gate.

Flotox, then, provides a simple, reproducible means for both programming and erasing a memory cell. But



4. Good endurance. The endurance of the EE-PROM depends on the threshold-voltage difference between the charged and discharged states. Though repeated cycling degrades thresholds, the chip should stay within tolerable limits for 10^4 to 10^6 cycles.

what about charge retention and refresh considerations with such a thin oxide? The key to avoiding such problems is given in Fig. 3, which shows the exceedingly strong dependence of the tunnel current on the voltage across the oxide. This is characteristic of Fowler-Nordheim tunneling.

The current in Fig. 3 rises one order of magnitude for every 0.8-V change in applied voltage. If the 11 orders of magnitude requirement is recalled, it is apparent that the difference between the voltage across the tunnel oxide during programming and that during read or storage operations must be in excess of 8.8 V.

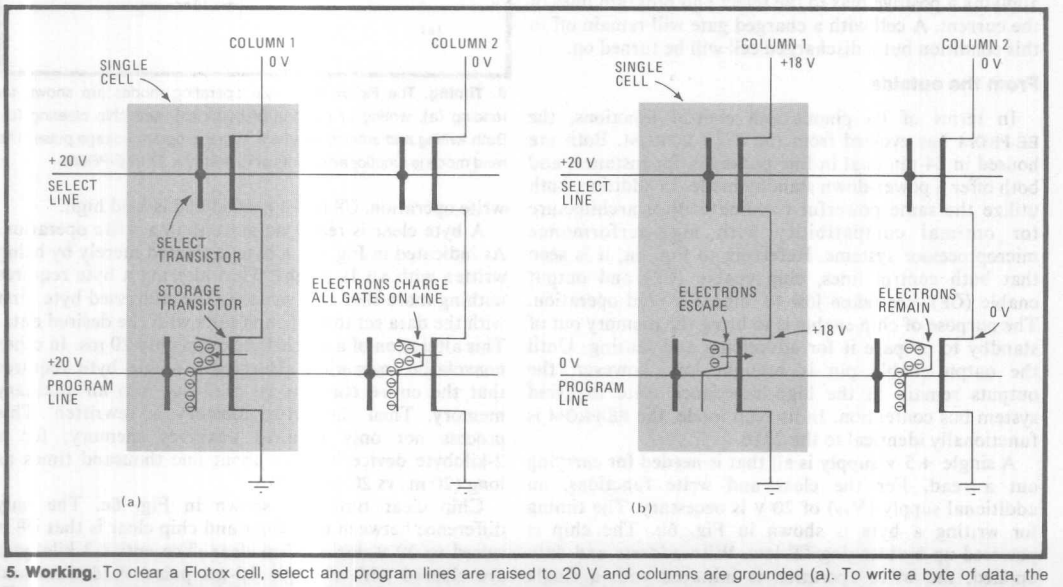
This value, including margins for processing variations, is reasonable. Furthermore, data is not disrupted during reading or storage so that no refreshing is required under normal operating or storage conditions. Extensive experimental testing has verified that data retention exceeding 10 years at a temperature of 125°C is possible.

Another important consideration is the behavior of the electrically erasable memory cell under repeated program erase cycling. This is commonly referred to as endurance. The threshold voltage of a typical Flotox cell, in both the charged and discharged states, is shown in Fig. 4 as a function of the number of programming or erasing cycles. There is some variation in the threshold voltages with repeated cycling but this remains within tolerable limits out to very high numbers of cycles—somewhere between 10^4 and 10^6 cycles.

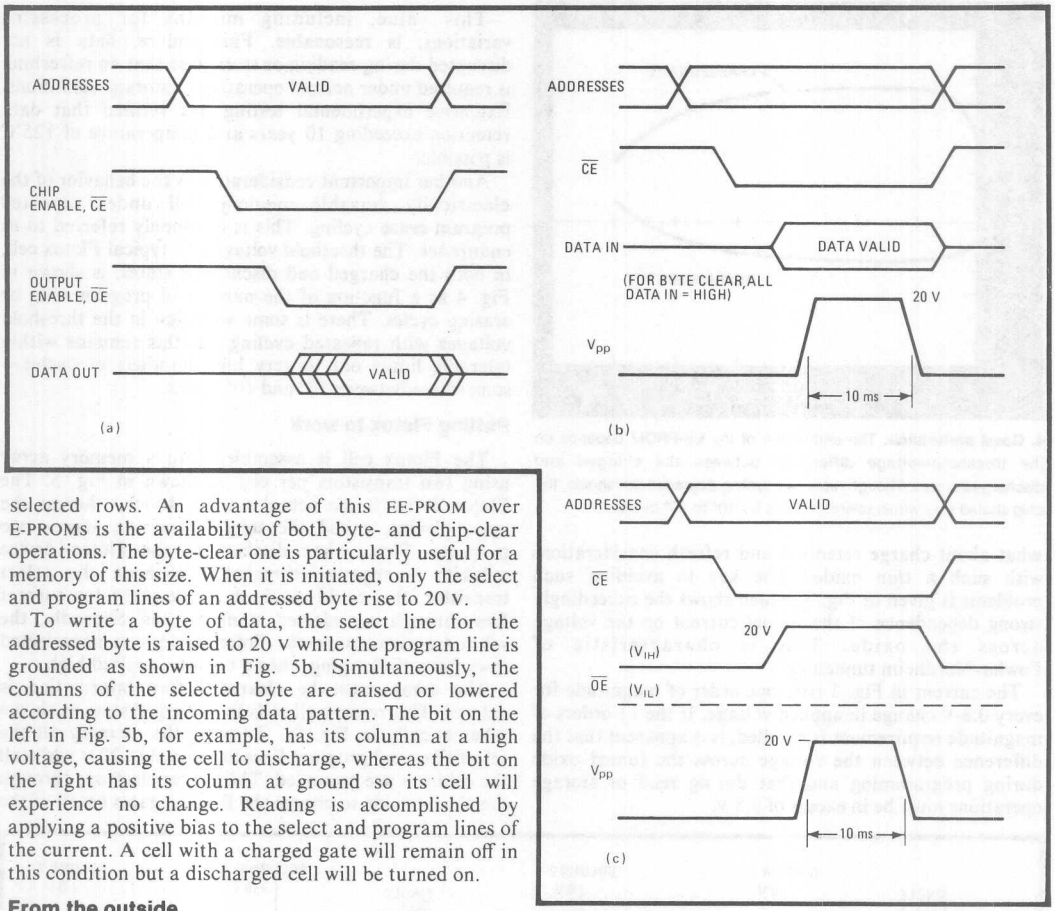
Putting Flotox to work

The Flotox cell is assembled into a memory array using two transistors per cell as shown in Fig. 5. The Flotox device is the actual storage device, whereas the upper device, called the select transistor, serves two purposes. First, when discharged, the Flotox device exhibits a negative threshold. Without the select transistor, this could result in sneak paths for current flow through nonselected memory cells. Secondly, the select transistor prevents Flotox devices on nonselected rows from discharging when a column is raised high.

The array must be cleared before information is entered. This returns all cells to a charged state as shown schematically in Fig. 5a. To clear the memory all the select lines and program lines are raised to 20 V while all the columns are grounded. This forces electrons through the tunnel oxide to charge the floating gates on all of the



5. Working. To clear a Flotox cell, select and program lines are raised to 20 V and columns are grounded (a). To write a byte of data, the program line is grounded and the columns of the selected byte are raised or lowered according to the data pattern (b).



6. Timing. The Flotox memory's operating modes are shown for reading (a), writing or clearing of bytes (b), and chip clearing (c). Both writing and erasing require a 10-ms program-voltage pulse. The read mode is functionally identical to that of a 2716 E-PROM.

selected rows. An advantage of this EE-PROM over E-PROMs is the availability of both byte- and chip-clear operations. The byte-clear one is particularly useful for a memory of this size. When it is initiated, only the select and program lines of an addressed byte rise to 20 V.

To write a byte of data, the select line for the addressed byte is raised to 20 V while the program line is grounded as shown in Fig. 5b. Simultaneously, the columns of the selected byte are raised or lowered according to the incoming data pattern. The bit on the left in Fig. 5b, for example, has its column at a high voltage, causing the cell to discharge, whereas the bit on the right has its column at ground so its cell will experience no change. Reading is accomplished by applying a positive bias to the select and program lines of the current. A cell with a charged gate will remain off in this condition but a discharged cell will be turned on.

From the outside

In terms of its pinout and control functions, the EE-PROM has evolved from the 2716 E-PROM. Both are housed in 24-pin dual in-line packages, for instance, and both offer a power-down standby mode. In addition, both utilize the same powerful two-line control architecture for optimal compatibility with high-performance microprocessor systems. Referring to Fig. 6a, it is seen that both control lines, chip enable ($\overline{CE}$) and output enable ($\overline{OE}$), are taken low to initiate a read operation. The purpose of chip enable is to bring the memory out of standby to prepare it for addressing and sensing. Until the output-enable pin is brought low, however, the outputs remain in the high-impedance state to avoid system bus contention. In its read mode, the EE-PROM is functionally identical to the 2716.

A single +5-V supply is all that is needed for carrying out a read. For the clear and write functions, an additional supply (V_{PP}) of 20 V is necessary. The timing for writing a byte is shown in Fig. 6b. The chip is powered up by bringing $\overline{CE}$ low. With address and data applied, the write operation is initiated with a single 10-ms, 20-V pulse applied to the V_{PP} pin. During the write operation, $\overline{OE}$ is not needed and is held high.

A byte clear is really no more than a write operation. As indicated in Fig. 6b, a byte is cleared merely by being written with all 1s (high). Thus altering a byte requires nothing more than two writes to the addressed byte, first with the data set to all 1s and then with the desired data. This alteration of a single byte takes only 20 ms. In other nonvolatile memories, changing a single byte requires that the entire contents be read out into an auxiliary memory. Then the entire memory is rewritten. This process not only requires auxiliary memory; for a 2-kilobyte device it takes about one thousand times as long (20 ms vs 20 seconds).

Chip clear timing is shown in Fig. 6c. The only difference between byte clear and chip clear is that $\overline{OE}$ is raised to 20 V during chip clear. The entire 2 kilobytes are cleared with a single 10-ms pulse. Addresses and data are not all involved in a chip-clear operation. □

Application Briefs 3

April 1981

A Variable Attribute C.R.T. Terminal

John F. Rizzo
Special Products Division
Applications Engineering

AFN-01913A

A VARIABLE ATTRIBUTE CRT TERMINAL

As the proliferation of the microcomputer continues, there will be an ever increasing requirement for local CRT terminals in households, businesses, and offices. This proliferation is expected to occur throughout the world, which places a burden on the terminal designer to accomodate a wide variety of languages, processing speeds, and transmission protocols. Given memory elements and tools available today, it is difficult to achieve a cost-effective design that will deal with all these variables. The 2816 offers an excellent alternative in the design of the CRT controller by allowing a high degree of universality and a virtually unlimited number of terminal attributes.

The E^2 family offers an excellent alternative to the system designer for use of a non-volatile Electrically Erasable memory device. The 2816 can contain both the raw program needed by the CRT terminal to perform basic functions, in addition to storage of the parameter information needed for local configuration. Some of the information that can be contained is baud rate transmission information, configuration of the terminal information such as parity detection reverse video, and full or half duplex modes. 2816s can contain these fundamentally basic constants which can be updated in the field by the user. This removes all of the switching components required in the past, and adds a higher degree of manufacturability and reliability to the terminal design.

In addition, the 2816 can be used as a look-up table for specific character fonts or graphic generation capabilities. This allows the terminal manufacturer to configure the font and language characteristics after manufacture, before shipment. For example, if a specific terminal is going to be shipped to a Far-Eastern nation, the font characters for that typeset can be programmed into the 2816 and shipped to that particular country. Another alternative is to allow programming of the font characters locally at the final destination of the terminal. The user can then program specific fonts and characters as required.

Even greater flexibility is possible from the graphics generation standpoint. It is simple for a user to place the terminal into a graphics mode and generate special graphics characters unique to the application. This can occur through local configuration of graphic types. The terminal could have a graphics mode, where a basic map of the character is presented on the monitor. The user then locates inside the graphic boundaries the necessary information he wishes to display. After this special graphics character is composed, the user simply pushes a command key on the terminal which loads that graphic character into E^2PROM. This is an extremely powerful application for the device because it allows each user to fit the particular terminal to a particular application. Scientific users can construct scientific or calculational characters and fonts, while businesses can configure business- or table-oriented fonts.

The block diagram of the system indicates that is used 2816 as a character generator store. The microprocessor used could be a high-speed 8086-2, or perhaps a 8088 microprocessor. Within the system is a 8279 keyboard display controller, which is used to interface with a standard terminal keyboard. In addition, we can use an Intel 8275 or 8276 CRT controller to generate graphic information on the face of the CRT. Also local to the system is an E^2 controller which is used to interface the 2816 to the microprocessor.

Other than the basic components within the system, we may wish to add a serial I/O interface which will allow remote configuring of the characters and communication protocols. The terminal can have a serial load operation where the 2816 is updated after receiving a command character. Other than the basic components, much of the functional operation of the terminal is determined by software.

The 2816 adds the capability of custom graphics, user-definable fonts and character sets, and programmable communication protocols. All this is possible because of the capabilities that the E^2 brings to system designs.

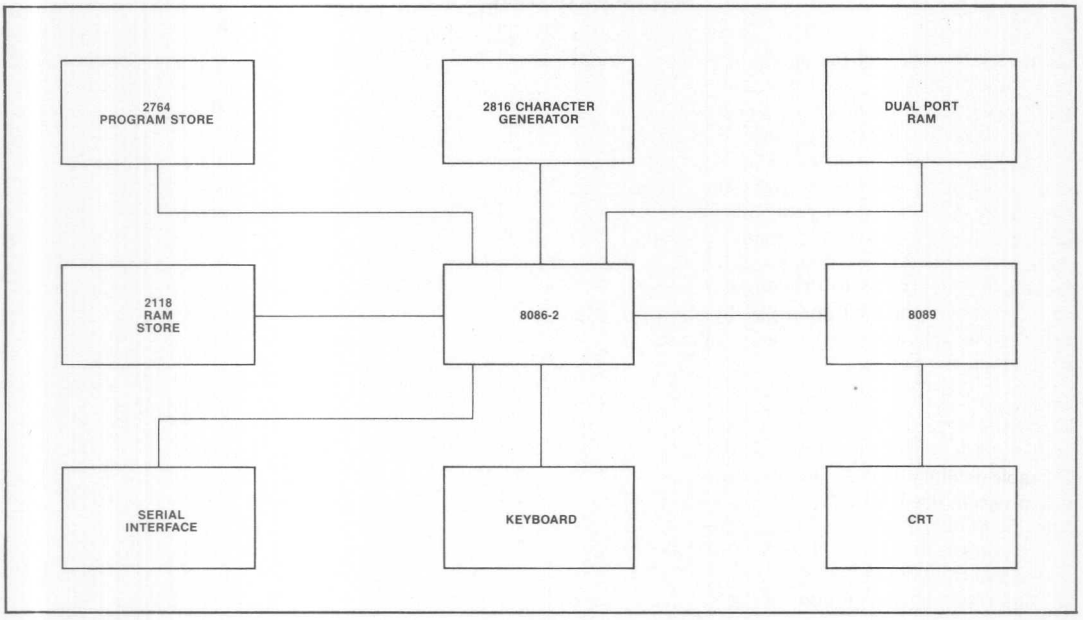
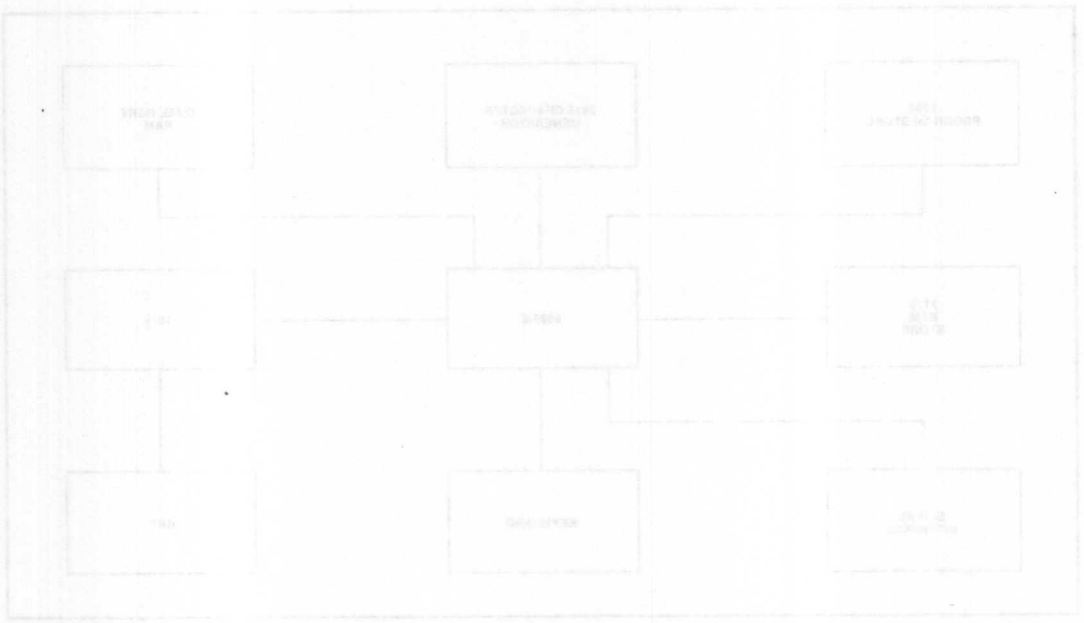


Figure 1. Variable Font C.R.T. Terminal



April 1981

Point of Sale Terminal

John F. Rizzo
Special Products Division
Applications Engineering

POINT OF SALE TERMINAL

Remote reconfiguration capability—that design feature can save millions of dollars in Point Of Sale Terminal service costs. With the capability of Intel's 2816 Electrically Erasable PROM, remote changes in terminal constants are now possible—no service personnel are necessary. How often have product codes and pricing information needed changes? In today's economy, one might answer "too frequently". With service costs today of over $100 per hour, those changes can be very expensive. The 2816 benefits users of Point of Sale Terminals by eliminating service costs. In this application brief the system architecture and user benefits of a 2816-based terminal will be discussed.

Point of Sale Terminals typically use look-up tables to contain product descriptions and pricing information. These tables require several different characteristics to operate optimally in a point of sale environment. The first storage attribute is non-volatility; look-up table data must be held without power for many months or years. Secondly, a dense storage media is required because typically many products with complex encoding schemes are loaded into the look-up tables. Finally, a media that can be changed relatively easily is needed because pricing and product information changes frequently. All of these necessary features have been satisfied in the past with EPROM memory, or CMOS RAM with battery backup.

Unfortunately, these media have drawbacks. EPROMs, while low cost, dense, and non-volatile, cannot be changed in the field without the use of a service technician. CMOS and battery backup offer more flexibility at a lower density, but can suffer reliability problems if the battery and backup system aren't properly designed. The 2816 E^2PROM from Intel offers users all the characteristics of EPROM with the flexible advantages of battery backed up RAMs. Look-up table data can be stored non-volatily, but can be changed while in system. Figure 1 shows the block diagram for such a system. The terminal is composed of a high-performance microcomputer, such as the 8051. In addition, 2816 memory is used as data and as look-up table storage. The typical I/O device structure for a terminal also exists in the system as shown. The most important interface indicated on the block diagram is the serial I/O link. This datacom or telecom link provides the system with remote reconfiguration capability. The contents of the 2816 can be changed from a central location, without need for costly human service.

The look-up table contains product description and pricing information. Once the table has been written, the CPU can read from it as necessary to translate product entry codes to price information. If for some reason the table data needs to be changed for pricing or product updates, then the central computer simply sends update commands and new data to the remote POS processor. Since all remote terminals are linked together at a central location and are in periodic communication with other, such an update can occur as a part of normal inter-processor communication.

The in-system erase capability of 2816 memory allows the table data to be changed remotely, while preserving the stand alone nature of the terminals. Without E^2 capability, a service technician would be required to change the table data.

In addition to containing product description and pricing data, the 2816 can store special data unique to a particular location. If a set of locations within the memory is set aside for reorder codes, then as a location runs short of a particular item, the computer can automatically restock it. If particular information is sensitive, the 2816 can store encryption codes and software lockout mechanisms.

Another capability gained from the use of E^2 memory is that daily totals in sales volume and product quantities can be stored in the 2816 memory. This information can be accessed by both the local users as well as the central data bank.

To summarize, in the 2816-based P.O.S. terminal described here, flexibility and greatly reduced service costs are the key. The E^2 memory contains product information that can now be changed from a central location without the use of very costly service personnel. The 2816 yields an ideal solution to data table storage problems in frequently altered point of sale systems.

AFN-01913A

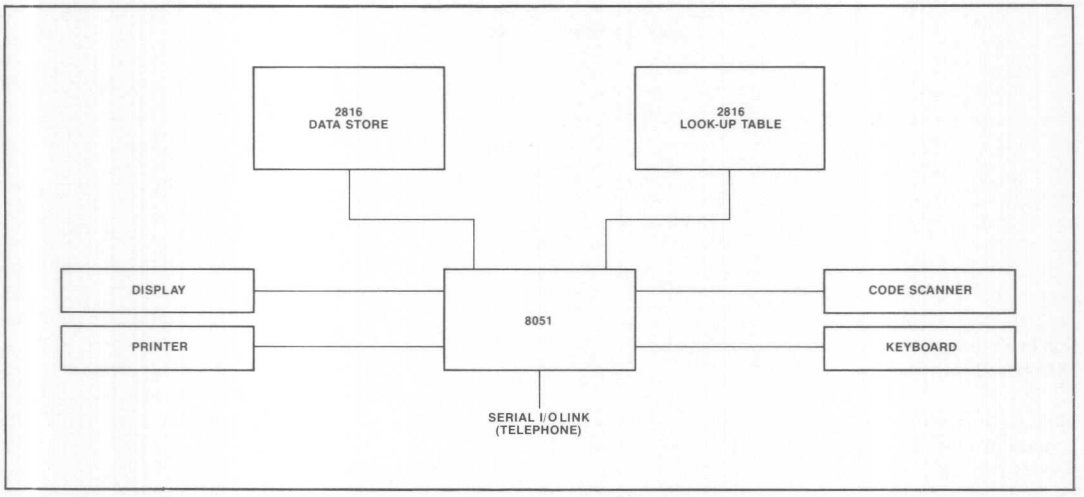


Figure 1. Point of Sale Terminal

August 1981

2816 E^2PROM Eliminates PROM Programmer Obsolescence

John F. Rizzo
Special Products Division
Applications Engineering

AB-3 INSTRUMENTATION: 2816 E^2PROM ELIMINATES PROM PROGRAMMER OBSOLESCENCE

How often are you unable to program a vendor's new PROM because programmer personality modules and software is unavailable? And how often as a programmer vendor do you lose customers because the latest programming algorithms do not exist in your machines? If the answers to the questions above are ever "too frequently," then Intel has a solution.

If a programmer vendor could eliminate hardware redesign and tooling, and a user could eliminate costly purchase of socket adaptors—would such a programmer design be attractive? If so, the Intel 2816 E^2PROM provides a system level solution. Due to the capability of the 2816 to be erased and written electrically while soldered to a Printed Circuit Board, PROM programming systems can be free of hardware redesign and socket adaptors. The 2816 can contain both pinout and algorithm data for a universal PROM socket. When a new device becomes available—a simple code change is all that is required to update the system. With such capability, the machine can operate and be revised nearly indefinitely—thereby eliminating obsolescence.

In this Application Brief, the concept of a Programmable PROM Programmer will be discussed. The basic architectural system structure will be detailed in general, then several implementation options will be discussed specifically.

SYSTEM ARCHITECTURE

Imagine the basic architecture of such a programmer system—the user must have the capability to easily adapt the program to suit a new device or modification of an existing algorithm. In addition, the programmer itself must be flexible enough to accept a wide variety of vendors, device types and unique algorithms. The most straightforward approach, but not the simplest, is the concept of a universal socket that is programmable through software. In such a system, each pin on the socket can be software configured to perform any function. Now, when a new device or algorithm becomes available, the software need only to change—no hardware modification is necessary. The physical constraints required on the system in many ways will dictate what type memory device is appropriate.

First, the area of memory used to contain the pinout and electrical algorithms must be non-volatile—when the machine is shut off, it must not lose the critical device data. Second, since the software will be permanently imbedded into the equipment, the data must be able to be altered electrically. Third, the memory used must be low cost—the value it presents the vendor in socket adaptors or tooling must be offset by the cost it adds to the equipment. Fourth, only those locations containing new algorithm data, or modifications to existing data, need be changed. If the entire memory contents are reset, all pinout data must be reloaded—with room for error and device programming problems. Finally, the memory must be dense enough to hold a wide variety of algorithm data, and fast enough to execute high speed programming algorithms.

The 2816 E^2PROM from Intel satisfies all of the above system requirements: 1) It is completely non-volatile, 2) it can be written and erased electrically while soldered to a printed circuit card, 3) the 2815 is priced at less than $20 in quantity—making it highly cost effective, 4) the device allows byte erase operations—only those locations that need to change can change, 5) its 2KX8 density, and 6) 250 ns access time make it dense and fast enough for most applications.

BLOCK DIAGRAM

To design such a system, let us first examine the general architecture. A conceptual block diagram is shown in the figure below (Fig. A). An Intel 2764 EPROM is used to store the basic operating software that the programmer runs on. A byte wide static RAM, the 21812, is used for scratch pad and temporary data storage. 2816 is used on both pinout and programming algorithm look-up tables. Digital to Analog converters are used to interface the digital MPU commands to the programmed PROM's analog requirements. I/O is used to instruct the machine how to operate, and allows modification of E^2 memory.

The overall operation of the programmer would go as follows: The user selects a device type and pushes a few keys on the I/O device. The MPU translates those keys into a device pinout and programming algorithm type. This programming data is selected from the 2816 look-up tables. The pinout is then configured at the universal socket. The device is then programmed according to the programming look-up tables. Should a new device become available, then the user need only to modify the E^2 look-up table to allow the device to be programmed.

LOOK-UP TABLE MODIFICATION

Modifying the look-up table can be done in several different ways. Changing parameters through the keyboard is an option—but has the disadvantage that the data could be entered incorrectly, or other portions of the look-up table could be changed inadvertently.

Linking the machine to a central data computer through a modem is another option that is probably too expensive for both end-user and programmer manufacturer. Using magnetic paper tapes or cards is another possibility, but now additional I/O devices are required. The option that seems the most effective is to use the programmer itself to modify its look-up table.

In this case the programmer manufacturer need only send a low cost EPROM to the user. The user would simply place the EPROM in the programmer, placing the programmer into a "Learn" mode. This special mode would simply copy the contents of the EPROM into the E^2PROM, free from user error or inadvertent tampering. Because the EPROM is so low in cost, the user would simply discard it after use. A low density EPROM could be used (such as a 2716) because only a small portion of the E^2 storage area needs to be changed. Figure B shows the possible memory map of the "Loader" PROM. Pointers are used for address and data and only those locations needing to be changed are modified in the E^2 memory.

CONCLUSION

Dealing with changing vendor algorithms and rapidly advancing technology can be a difficult task for both PROM programmer vendors and users alike. The 2816 E^2PROM from Intel can eliminate problems associated with the above, through the use of software controlled pinout and programming algorithms. Changes to operating look-up tables can be done simply, easily, and error free. E^2 benefits users with more up-to-date programming systems with reduced adaptor costs. E^2 benefits programmer manufacturers through reduced hardware tooling, faster algorithm changes, and more rapid time to market entrance. Generally, it is an excellent and cost effective addition to PROM programmers of today.

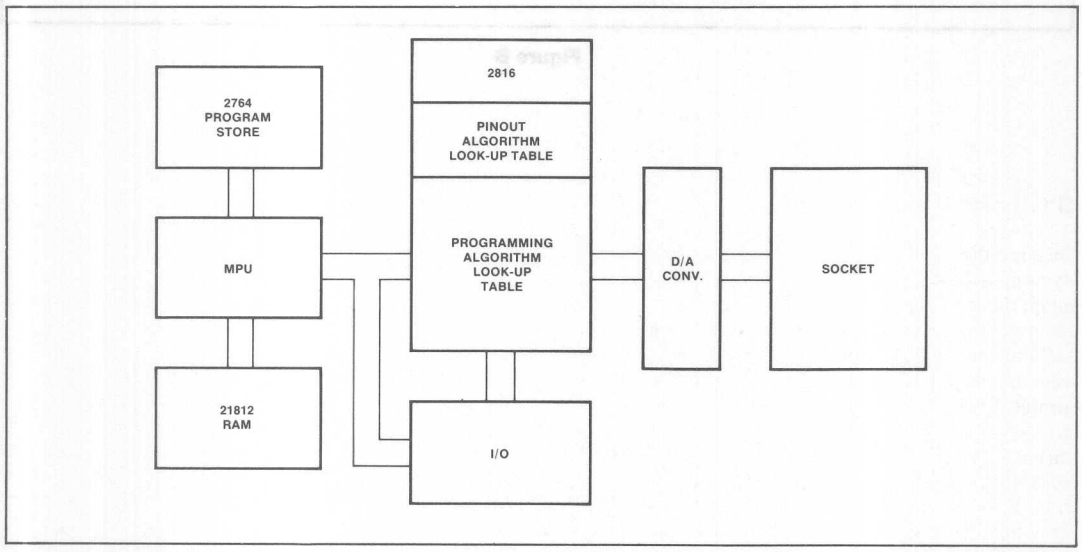


Figure A

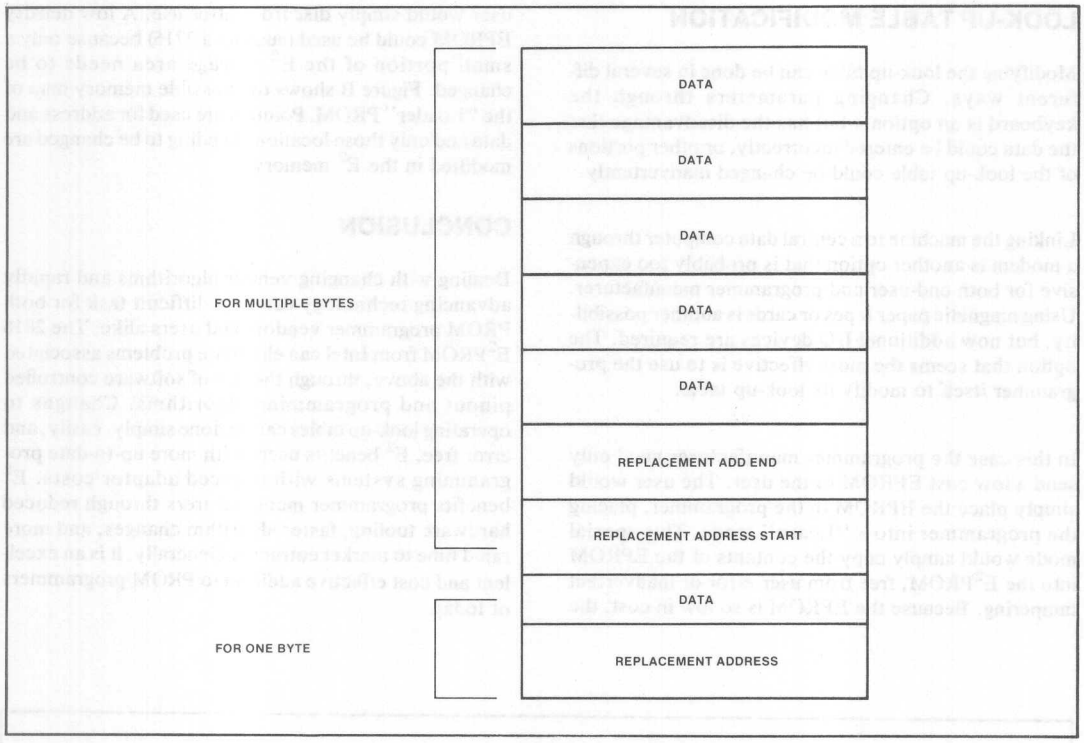

DATA
DATA
DATA
DATA
DATA
REPLACEMENT ADD END
REPLACEMENT ADDRESS START
DATA
REPLACEMENT ADDRESS
FOR MULTIPLE BYTES
FOR ONE BYTE

Figure B

August 1981

ROMs Become Flexible with 2816 E^2 PROMs

John F. Rizzo
Special Products Division
Applications Engineering
Intel Corporation

Kirk Wimmer
Design Engineer
Tektronix Incorporated

Reprinted from *EDN*, April 29, 1981.

Users of ROMs for program storage in microprocessor based systems gain the advantage of very dense, very low cost program storage. But how often are these advantages lost when a ROM code is incorrect, or not optimized for a given customers needs? When a software bug is discovered, or an end user wants a slight program modification, ROMs must be discarded to be replaced with modified versions. If many systems undergo such a change, literally thousands of ROM's are wasted. Intel has an excellent solution to ROM modification problems that gives the end user the best of both worlds—low cost and high code flexiblity. The 2816 E^2PROM, from Intel, used as a front end ROM patch allows bugs to be easily removed, and code changes simply made—without costly throwaway of incorrect Read Only Memories.

The memory systems employing both ROM and E^2PROM gains benefits of both, through a technique called ROM patching. A ROM patch is essentially a replacement (through a software, or hardware branch) of existing ROM code. When the microprocessor requests data from a ROM that is incorrect, the E^2 patch system recognizes the bad address, disables the ROM and outputs the modified or correct data that is stored in E^2 memory. If a entire section of code is bad, then the E^2 memory is simply "patched in" through subroutine branches. Because the branch routine need only be a relatively small portion of code, the economics of dense, low cost ROMs can be supplemented with the flexibility of higher cost, less dense E^2PROMs. The dramatic flexibility of E^2PROMs is gained at nearly the same low cost as ROM.

E^2PROM is used to advantage because now ROM patched can be made through the CPU's I/O systems. New patch data can be loaded into the E^2 electrically, without having to disassemble the equipment and swap EPROMs or PROMs. To extend the concept even further, these ROM patches can be made from a remote service processor through a data-com or telecom link. No service personnel would be necessary.

The 2816 offers features ideal for such an application. Its access time of 250ns is necessary to allow fast patching without slowing down the CPU. Because only certain bytes of information would change, the capability for individual byte erase is necessary.

A heavy user or ROMs, Tektronix, of Beaverton, Oregon, is now using the ROM patch technique. They currently use EPROMs as the patch medium, but E^2 offers a simple and more optimum use of the technique.

Attached is a reprint of an article written by Tektronix that discusses in great detail the technical implementation and advantages of such a scheme.

Field-programmable patches simplify firmware maintenance

The need to modify design firmware is virtually inevitable. Of several possible approaches to the job, one can offer higher efficiency and greater ease of use than the rest.

Kirk Wimmer, Tektronix Inc

You can use the ROM-modification technique developed in this article to easily and efficiently patch the firmware for your μC-based designs. And the users of those designs can also employ the method—without resorting to a service call. The technique overcomes most of the shortcomings of traditional firmware-modification methods, also discussed here.

The need for firmware modification arises from several sources. Implementation errors (bugs in the code) might be present, for example, or your initial design might exhibit specification errors. More importantly, though, your design will very likely require modification over its lifetime to accommodate expanded features and new options.

Planning for such changes is thus absolutely necessary. But actually changing ROM masks is an unacceptable approach because of its expense, long leadtimes and minimum-order-size requirements. Thus, you need an efficient way of fooling your design's μC into sensing the required firmware changes even though those changes are not actually made in the system's masked ROM. Several methods of performing this task exist.

Try a transfer table

One such method, the transfer-table approach, involves dedicating approximately 20% of the ROM's address space to the task of adding a patch PROM's contents. In this PROM, you store a transfer (jump) table that serves as a directory for the ROM's routines. Every time your system's software program calls a routine, it vectors to that routine via the transfer table. To correct a firmware error, therefore, you first write a new routine in the remaining PROM space. Then you alter the transfer table so the system gains access to the new PROM routine rather than the erroneous version in ROM. Because the amount of patch space available determines the scheme's correction capacity, the approach requires that you reserve adequate patch space for several software modules and that you partition the firmware into small modules.

Transfer-table patching has several advantages. It not only works for firmware coded in assembly language, but also applies to firmware written in a high-level language. That is, you needn't be concerned about the details of the microprocessor instructions in the area where the problem exists. Additionally, after you finalize the firmware code and complete all desired modifications, you can replace the patch PROM with a pin-compatible ROM to minimize cost.

The scheme also has two disadvantages, though. First, an error in a particular routine requires—at minimum—replacement of a module from the routine's entry point through the point of the desired change. In some applications, you might need to replace the entire module, rapidly consuming available patch space. Second, the patch PROM must always be part of the system, even if you make no firmware corrections, because the method always employs the transfer table.

Vary the transfer-table approach

In another ROM-patching scheme that's similar to the transfer-table approach, you initially group the firmware modules functionally and then distribute them into several small ROMs. Next, you construct a transfer table in RAM from information found in the ROM headers upon power-up. Alternatively, you can use a dedicated, fixed location in each ROM to house the transfer table for its modules. In either case, if a problem exists in one functional area, you need replace only that function's ROM. For that purpose, you can use a pin-compatible PROM until a new masked ROM becomes available. When you install the new device, the transfer table gets updated automatically.

A disadvantage of this approach is that you must provide expansion space in each ROM when locating the firmware modules. Furthermore, if you require more patch space than is available in a particular ROM, you

AFN-02022A

Transfer-table scheme reserves address space for a patch PROM

might have to change a second ROM to also utilize its patch space. And obviously, because it uses smaller ROMs, this approach consumes more space and power than the transfer-table scheme.

Field-programmable patch solves some problems

A more hardware-oriented approach to the problem of modifying a masked ROM's contents (**Fig 1**) utilizes a device termed a field-programmable ROM patch (FPRP). (See EDN, April 5, 1980, pg 88 for more information on these devices.) In this approach, the μP places addresses on the address bus during ROM read cycles, and the FPRP monitors those addresses. If the bus carries an address that the FPRP has been programmed to recognize, the FPRP's $\overline{\text{Flag}}$ output goes LOW, disabling the ROM and preventing it from putting its data on the 8-bit data bus. Instead, the FPRP's data outputs activate and supply the bus with a replacement byte. The μP interprets this byte as having come from the ROM. The strategy thus involves programming the FPRP to recognize addresses where erroneous ROM code exists; the FPRP in turn supplies the data bytes needed to correct the code.

You must often make insertions to existing ROM code without offsetting those insertions with deletions of corresponding size. To solve this problem, you can program the FPRP to substitute a Jump instruction to the address bus; the instruction's destination is a location in the patch PROM. You also write into the patch PROM the code slated for insertion, followed by another Jump instruction, which returns program flow back to ROM. However, because the FPRP has substituted a Jump for several bytes of ROM code, you must also duplicate the displaced bytes in the patch PROM.

You can see that the FPRP approach permits flexible firmware correction both on a byte-for-byte basis and in cases where an entire ROM routine needs modification. Its chief disadvantage is rapid consumption of FPRP capacity, because the FPRP outputs connect directly to the data bus. Furthermore, the approach uses one FPRP product term (with 16 word blocks per product term) for each byte it places on the bus. Consequently, when you need a Jump instruction, one product term serves the Jump opcode and two more serve the high and low address bytes. With only 48 product terms available, this ROM-patching configuration can thus soon become limited.

Another disadvantage of the FPRP method stems from the fact that changes occur in machine code at specific address locations, which are not readily accessible if you write the firmware in a high-level language. And if you require a 16-bit data bus, you need two FPRPs to drive the bus with the higher and lower data bytes.

An FPRP variation does the job

A variation on the FPRP approach overcomes most disadvantages. It was developed to modify firmware for Tektronix's Model 7854 oscilloscope and uses the FPRP to monitor a 15-bit address bus. But it doesn't connect the FPRP's outputs to the data bus (**Fig 2**). Instead, those outputs drive the seven most significant

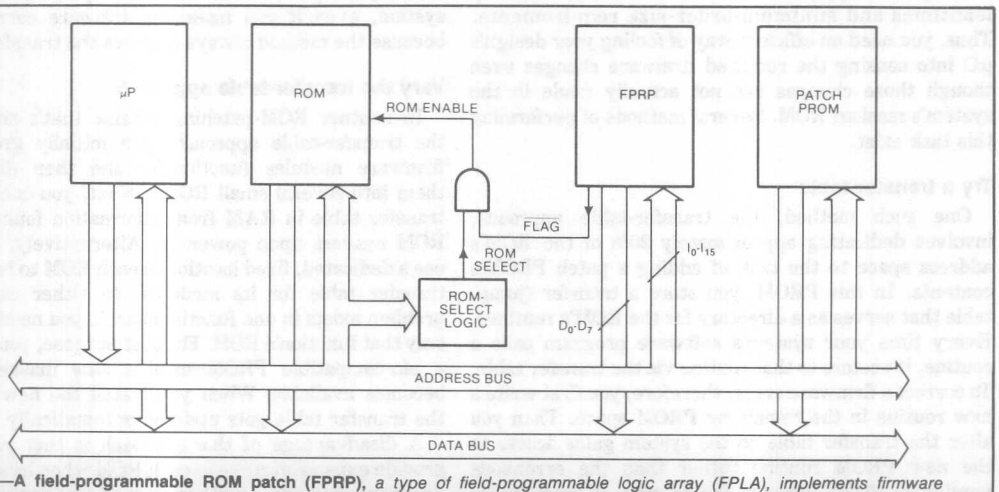


Fig 1—A field-programmable ROM patch (FPRP), *a type of field-programmable logic array (FPLA), implements firmware corrections in a memory system containing masked ROM. Its inputs connect to the system's address bus, its outputs to the data bus. Available in either open-collector or 3-state versions, the FPRP's outputs activate internally if and only if one of its input product terms is satisfied. Simultaneously, its* $\overline{\text{Flag}}$ *signal goes LOW, inhibiting the ROM's data outputs from reaching the data bus.*

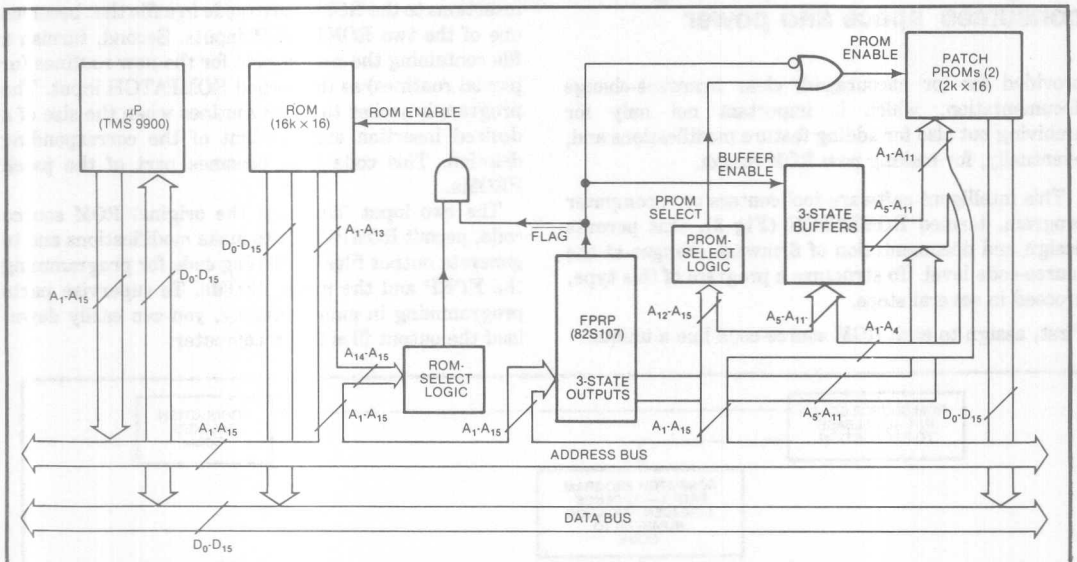


Fig 2—A variation of Fig 1's FPRP scheme, *this firmware-modification approach uses a field-programmable ROM patch to monitor the address bus. Instead of driving the data bus, though, the FPRP drives patch-PROM address lines A_5 to A_{11}. This strategy allows the FPRP to select a block of 16 PROM words, consuming only one FPRP product term during a data-read operation.*

address lines (A_5 to A_{11}) of two 2716 (2k×8) patch PROMs. The address bus directly drives the four least significant address lines (A_1 to A_4) of these PROMs.

This setup allows the FPRP to select a block of 16 words in the PROMs. Thus, unique data gets read while consuming only one FPRP product term. This read process takes advantage of the fact that you can program the FPRP inputs to the Don't Care state.

To employ the technique, you partition the two patch PROMs into two sections. The first contains 768 words (48 product terms×16 word blocks per product term), reserved for access only by the FPRP. The second stores the remaining 1280 words, which serve as patch space to hold replacement routines and code sections that in effect get inserted into the ROM. For added storage, you can install two more PROMs and obtain another 2048 words of patch space.

You can alter a single ROM word by programming the FPRP to recognize its address. You store the correct word in the appropriate location in the patch PROM. Several consecutive words might get corrected with only one FPRP product term by programming Don't Cares into the least significant inputs. All addresses affected, however, must fall within the same block of 16 words (ie, addresses A_5 through A_{15} common to each).

If a particular modification requires code insertion into the stream of instructions already in ROM, you can substitute a Branch instruction for two ROM-code words at the appropriate location. The code overlaid by this instruction and the code to be added can all reside in the patch-PROM area. You can address this area as ordinary memory without FPRP intervention. However, you must follow the added code with an instruction to branch back to ROM.

You can usually implement a 2-word Branch-instruction substitution of this kind with only one FPRP product term. The determining factor is whether or not address bits A_5 through A_{15} are identical for the two address locations where the branch substitution occurs. In the 7854's firmware, two FPRP product terms always get consumed to correct checksum and header information—a process that leaves 46 product terms available to fix firmware problems. Chances are thus one in 16 that a particular correction requires two product terms rather than one. Therefore, the 7854 implementation of this approach permits 40 or more separate corrections with better than a 93% confidence level.

An efficient software tool

Once you've recognized the necessity of providing for firmware modification and chosen a particular method, carefully consider the procedures required to actually make the changes. In the 7854's design process, for example, questions arose regarding how to program code into the FPRP and the patch PROMs. Hand coding proved undesirable because of its complexity and high error probability. Instead, an "intelligent" software tool implemented the desired firmware changes in hardware, easily and with minimum error. The tool also

Using small ROMs for patching consumes space and power

provided for (or encouraged) clear firmware-change documentation, which is important not only for archiving but also for adding feature modifications and, eventually, for making new ROM masks.

This intelligent software tool centers on a computer program, termed ROMPATCH (**Fig 3**), that permits design and documentation of firmware changes at the source-code level. To structure a program of this type, proceed in several steps.

First, assign to each ROM source-code line a unique identification number, used to describe deletions and insertions to the ROM source code in a file that becomes one of the two ROMPATCH inputs. Second, furnish a file containing the source code for the new routines (or partial routines) as the second ROMPATCH input. The program branches to these routines when the size of a desired insertion exceeds that of the corresponding deletion. This code then becomes part of the patch PROMs.

The two input files, plus the original ROM source code, permit ROMPATCH to make modifications and to generate output files containing code for programming the FPRP and the patch PROMs. To supervise parts programming in manufacturing, you can easily download the output files into a computer.

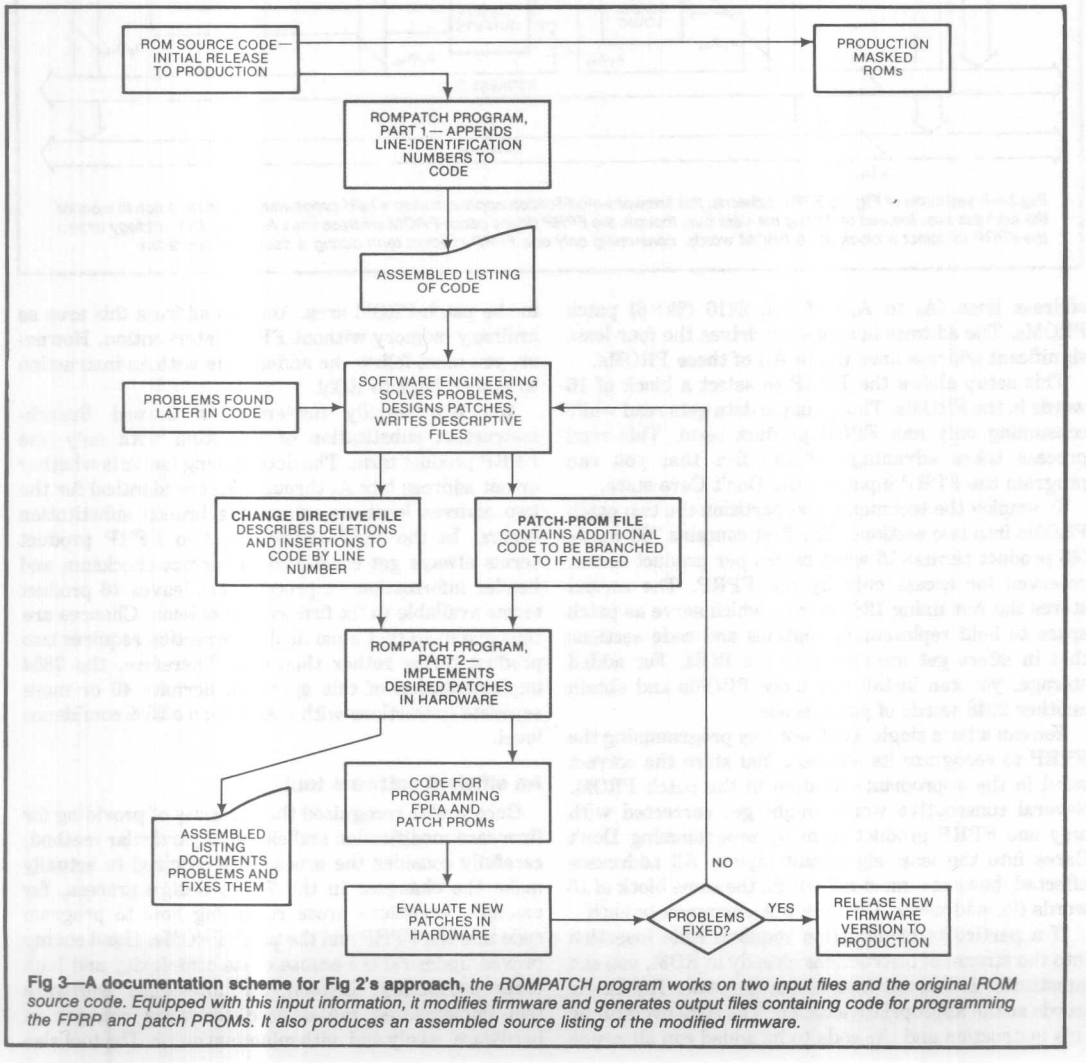


Fig 3—A documentation scheme for Fig 2's approach, *the ROMPATCH program works on two input files and the original ROM source code. Equipped with this input information, it modifies firmware and generates output files containing code for programming the FPRP and patch PROMs. It also produces an assembled source listing of the modified firmware.*

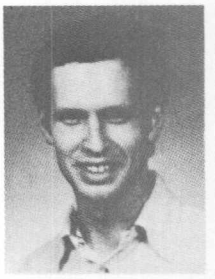
Field-programmable ROM patch monitors address bus for faulty code

Additionally, one of ROMPATCH's outputs produces an assembled source listing of the modified firmware. The deleted ROM code remains in this listing but gets transformed into comment statements that the assembler ignores. The inserted code and descriptive comment statements follow the deletion in this listing, documenting the nature of each problem and how it's fixed. This information, all in one place, proves important when you must make future modifications and order a new set of ROM masks.

An additional benefit of the ROMPATCH program allows configuration of power-up diagnostics to test the ROMs by first calculating checksums through their contents, then comparing those checksums with values stored in the headers. However, when you modify the firmware, the checksums are effectively calculated over altered data and thus change. To deal with this problem, ROMPATCH or a program like it must automatically substitute the correct value for comparison. As a result, the power-up test not only checks the ROMs, but also checks whether the FPRP and patch PROMs are working properly.

You must establish implementation procedures like the foregoing whether you select a hardware- or a software-oriented ROM-patching approach. Try to develop software tools that lessen the laborious task of generating code for programmable devices. These tools, of which ROMPATCH is an example, serve to reduce human error, promote complete modification documentation and, most importantly, fix firmware problems.

Author's biography



Kirk Wimmer is a design engineer in the Laboratory Oscilloscopes business unit at Tektronix Inc, Beaverton, OR. Currently project leader on a 7000 Series plug-in-unit design, he previously worked on ROM-patching techniques for the firm's Model 7854 Waveform Processing Oscilloscope. A graduate (BSEE and MSEE) of the University of Colorado, Kirk lists home remodeling and exploring alternative energy sources as his hobbies.

August 1981

Industrial Robots E² Shows Them the Way

John F. Rizzo
Special Products Division
Applications Engineering

AB-5 INDUSTRIAL ROBOTS: E²PROM SHOWS THEM THE WAY

Industrial manufacturing environments are changing. Once semi-skilled or unskilled labor occupied most of the manufacturing workforce. But now—as labor costs have skyrocketed—manufacturers are looking for new ways to reduce costs and exploit new technology.

One of the most revolutionary of these concepts is the use of robots for manufacturing processes. Robotics reduces manufacturing costs and increases worker productivity. The Intel 2816 E^2PROM can make robotics more efficient through the use of easily reprogrammable motion and action paths. In robotics systems when retooling is required, changes can be made simply, easily and cheaply by simply reprogramming E^2 memory. No physical system tampering is needed—changes in robot action are made through the cable, or through remote data link. E^2 memory offers excellent reliability and environmental capability in addition to high density and system flexibility, all needed for robotics systems.

The Intel E^2PROM is a 16K, byte or chip-erasable device that can be erased and rewritten electrically—in addition, the device retains data without power in a completely non-volatile fashion. In robotics systems, where mechanical and pneumatic devices are controlled by microprocessors, typically the robot's path is contained in a non-volatile memory device. The microprocessor sequences through the action coordinates and directs the path of the machinery. This action path typically is changed infrequently—changes only are made for retooling or for correction of program bugs. However, because these robots are becoming extremely versatile and are being used for many different applications, the need for a flexible program memory is becoming increasingly important. One system can then be designed for a whole host of applications and the end-user can then configure the system local to the application.

E^2 memory gives the end user this flexibility and allows robotics systems to be used in a wider variety of applications.

GENERAL SYSTEM ARCHITECTURE

The system design for such a controller in many ways is similar to existing designs. The main distinction is the added capability with E^2 memory. A block diagram for such a robotics controller is shown in Figure 1.

The main functional units are split into general processing, performed by a 10 MHz iAPX86, numeric data processing for rapid calculation motion data, Input/Output processing to control mechanical or electrical robotic elements, and vision capability. Numeric data processing is performed by an Intel 8087. The main system program store is contained in 2764 8KX8 EPROM. User's control the system through a CRT interface. The 2816 is used to control the system through a look-up table—all parameters that are flexible can be changed directly from the keyboard.

E^2/MICROPROCESSOR INTERFACE

A high-speed 16-bit MPU is used because of complex processing and numerical calculations necessary to support several degrees of freedom. Multiprocessing, in the form of 8089, and 8087, is used to offload tasks to more efficient processing elements. Because of the rate at which the CPU operates, relatively high-speed program and parameter storage is required. The 2764 and 2816 satisfies this need with read access times of 250 ns maximum.

In this system environment where E^2 is used as an infrequently changed parameter storage media, changes to E^2 memory can be made on-line. That is—the CPU can directly handle interface to the E^2 memory. No arbitration logic would be needed because the robotics would be in a retooling mode, the CPU could be dedicated to servicing the E^2 memory. Thus, the overhead logic would be minimal. A complete interconnect is shown in Figure 5.

E^2 MEMORY ADVANTAGES

The system's memory map would be partitioned in such a way to locate E^2 memory in the flexible portion of the memory space. This flexible memory could contain set-points, motion or control paths and other information that needs to be non-volatile but can still be changed dynamically while in system.

In addition to holding applications constant and parameters, the 2816 can contain program routines that would undergo change by the end-user. This essentially allows the users to custom fit the system to their particular application without having to require service from the equipment vendor. Such an approach reduces service costs and extends the useful life of the machine by minimizing obsolescence.

E^2 memory is also more cost effective than alternative forms of flexible memory. CMOS and battery systems require costly batteries, recharging and interface circuits and have serious environmental drawbacks. When all these factors are considered with replacement of battery, E^2 is extremely cost effective.

SUMMARY

In this Application Brief, we have discussed the benefits that E^2 memory gives to manufacturers of robotics and numerical control equipment. Flexible, non-volatile storage of parameters and software makes the system dramatically more attractive to users and reduces overall support required by vendors. All in all, the Intel 2816 is a perfect choice for Robotics systems.

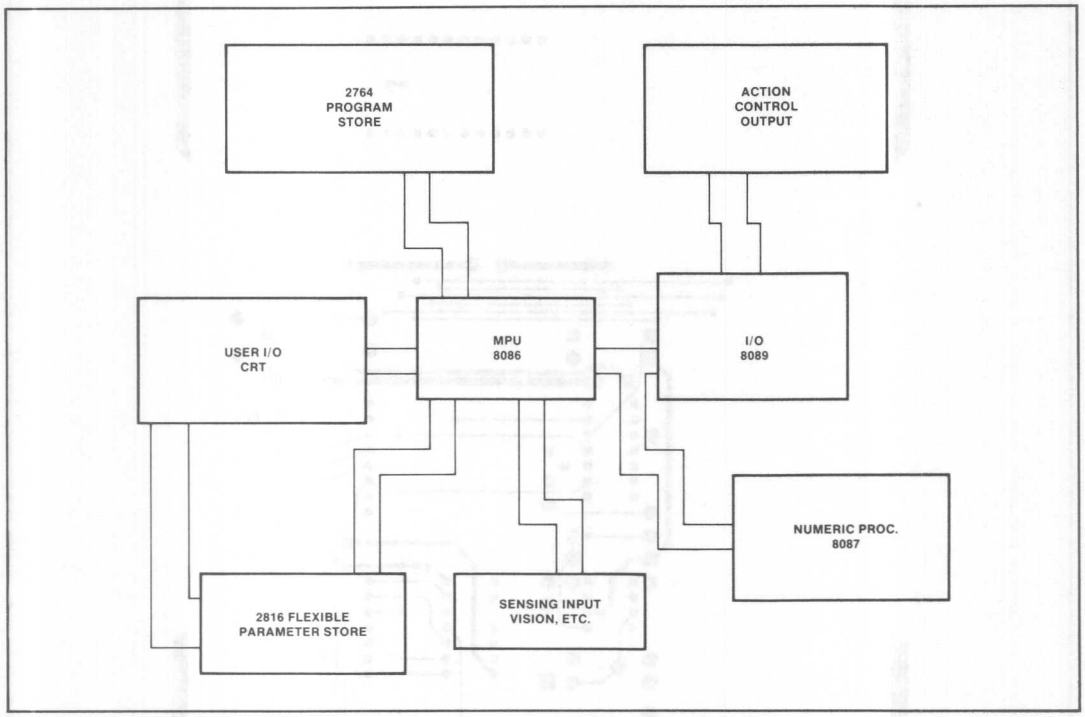


Figure 1

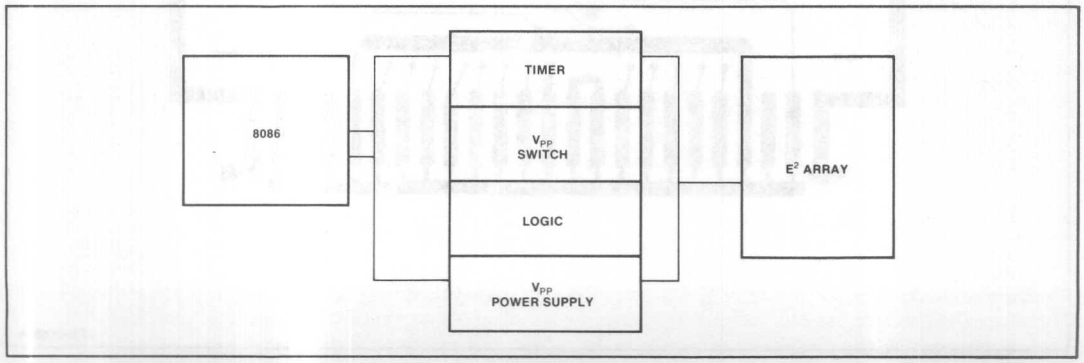


Figure 2

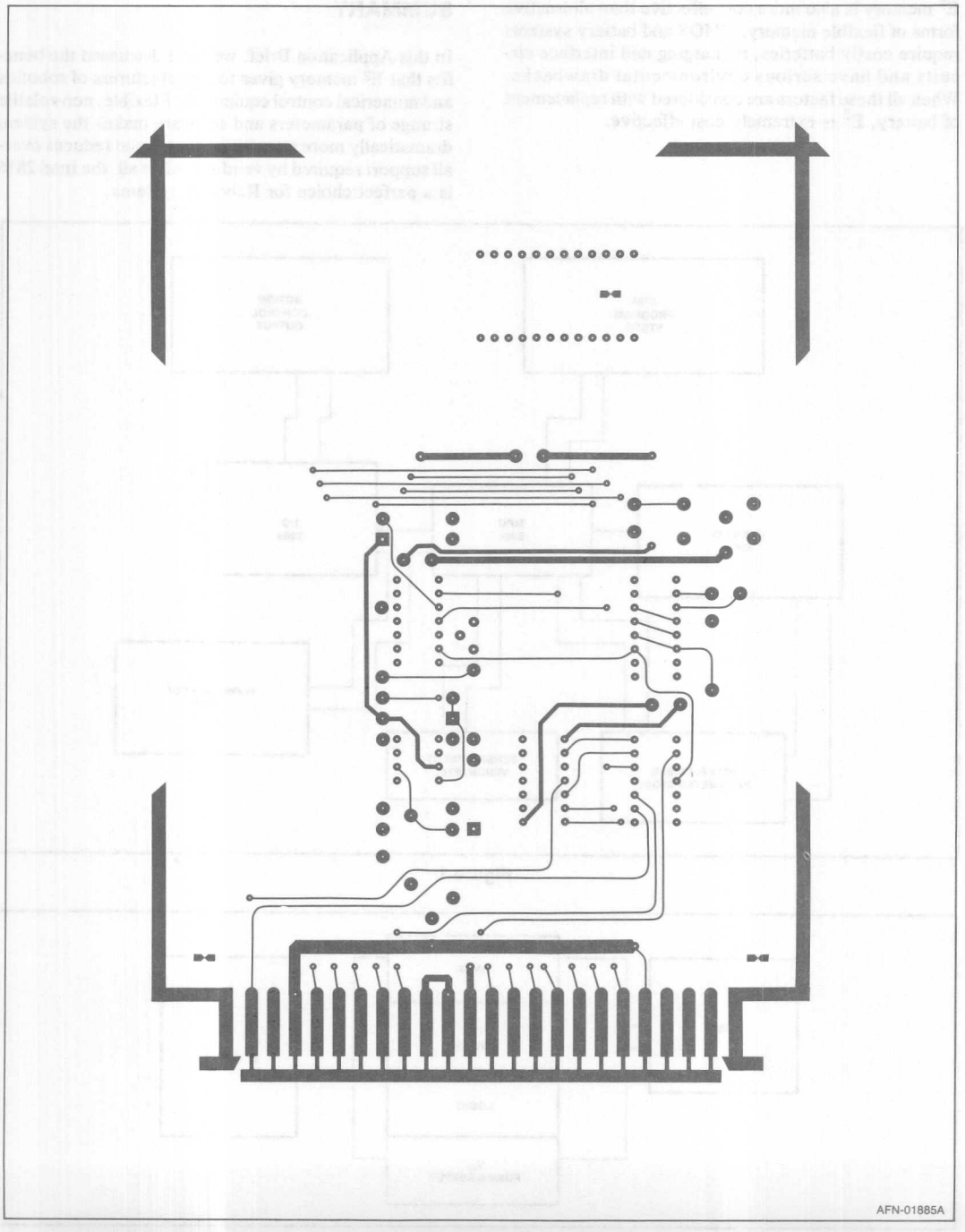


Figure 6c. E^2-Demo II Controller I (Continued)

October 1981

Prototyping with E^2 PROMs

Lawrence Palley
Special Products Division

©INTEL CORPORATION, 1981.

AFN-02024A

PROTOTYPING WITH E^2PROMs

The Intel 2815 and 2816 E^2PROM memory devices are being applied in many systems as a firmware storage medium. They give these systems a high degree of firmware flexibility in applications where firmware changes are a commonplace occurrence.

Many systems, once they are in the field, may not necessarily require the dramatic firmware flexibility which E^2PROMs provide. For these systems, EPROM or ROM memories are often chosen for firmware storage. Even on these systems, however, E^2PROMs offer unique opportunities for cost and time savings. Easier design and production start-up can often be realized by using E^2PROMs as prototyping vehicles and in early production runs.

E^2PROMs can be used cost effectively during the development of systems which will eventually utilize ROM or EPROM memories for firmware storage. This application brief describes how E^2PROMs can be used for firmware development prior to the actual production of ROM or EPROM memory boards. It describes the prototyping cycle and the resulting benefits E^2PROMs provide to prototyping applications —efficient use of engineering time, a faster prototyping cycle, and a faster time to get product to market.

The complexity of today's designs requires that much time and money be spent for the development of firmware. Several iterations of firmware will usually be needed before a system is debugged and production fully under way. Resulting costs can quickly mount if designers must do these firmware updates off-line from the testing process. The flexibility to change the firmware in these prototype systems, without replacing hardware or reprogramming PROMs, is provided most effectively by E^2PROMs.

The flowchart shown in Figure 1 illustrates a typical firmware prototyping cycle. In this cycle, firmware designs must wait for modifications to be implemented off-line. When a firmware error is detected, a new compilation of all codes must be implemented off-line, EPROMs or PROMs must be removed from the test system, erased, reprogrammed or replaced, and finally reinserted in their sockets before testing continues. During the period the update is occurring, testing is idled.

Figure 2 shows the same prototyping cycle, but with E^2PROMs used as the firmware storage medium. While developing a system with E^2PROMs, quickly identified firmware bugs can be immediately corrected. Firmware patches, which correct or route around the erroneous code, can be immediately inserted.

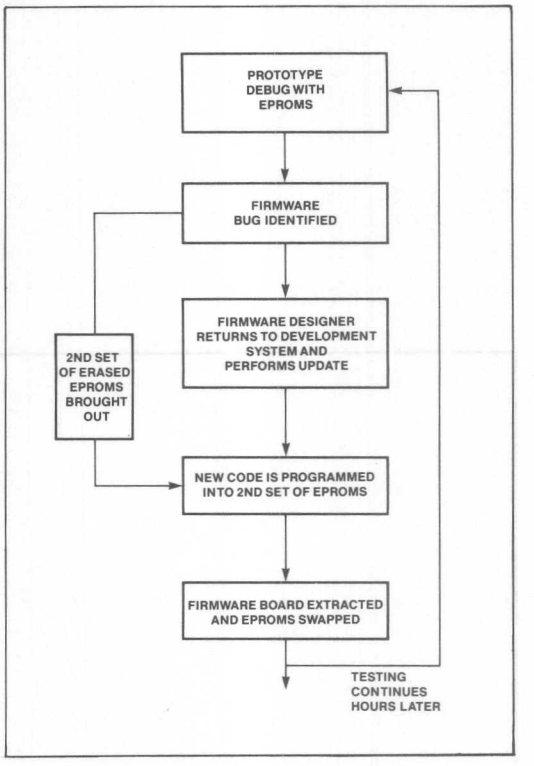


Figure 1. Typical Firmware Prototype Cycle

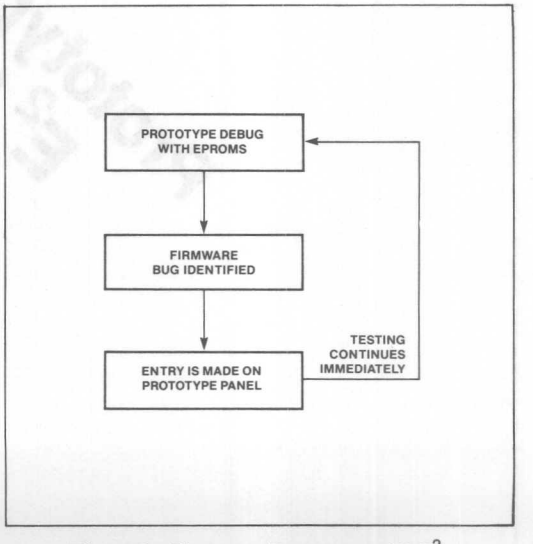


Figure 2. Firmware Prototype with E^2

Another case for E^2 usage during the prototype cycle is when a particular portion of code is not critical to the testing process. Often, it is desirable to delay testing of this code until critical code has been debugged first. Firmware branch code, which routes around this noncritical, untested code, can be inserted in the E^2PROM during initial testing. When the critical code is debugged, this branch can be removed, and the noncritical code debugged.

E^2PROMs offer much greater flexibility for prototyping than ROM, PROM, or EPROM alternatives. Additionally, because E^2PROMs are non-volatile, firmware is retained even when power is off—a significant cost savings over RAM prototypes, downloaded from disk. Continued downloads after every power down are not required.

Similar benefits are realized during a product's production start-up phase. Many times nonessential firmware for a system is not finalized until after production of the product is under way. By utilizing E^2PROMs, finalized firmware can be loaded quickly into the product after its assembly. No boards or ICs would need replacement for the new firmware. E^2PROMs would be used and programmed in-circuit.

IMPLEMENTATION OF AN E^2PROM PROTOTYPE

A big advantage of Intel's 2815 and 2816 E^2PROMs is their pin-for-pin read compatibility with the 2716 EPROM and other ROM ICs. Circuit boards, with sockets designed for 2816s, can easily accept 2716s when used in the read mode. This compatibility ensures that a firmware memory board needs to be designed only once for both prototyping and production. Discussed below is the design of such a board.

A major design consideration for the firmware memory board is that it has the minimum number of components, at the lowest cost, for production. For a board which is to contain ROM (e.g., it is used in a read-only mode alone after production), this necessitates that all circuitry used to program the 2815s or 2816 be located on a separate board for prototyping. This circuitry, which supplies the necessary timing, voltage, and waveshaping for programming the E^2PROMs, can easily be located on a small daughter board. This small board is easily connected to the memory board through a connector or cable. A block diagram of such a configuration is shown in Figure 3.

Operation of the programming daughter board is initiated by the chip enable and write enable signals. The chip enable input is derived from the E^2PROM address

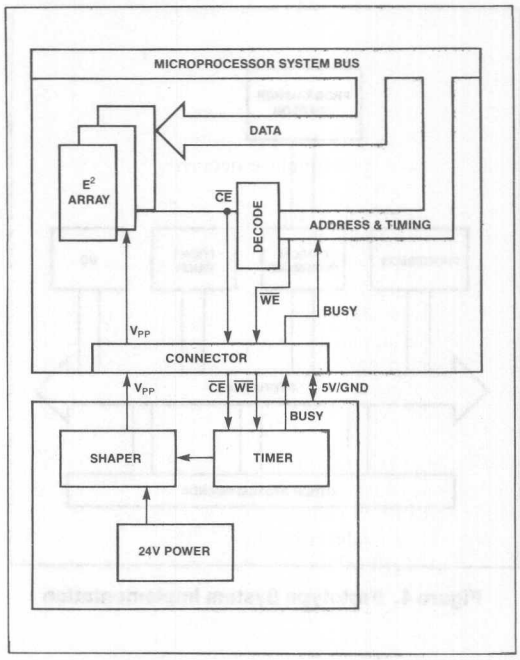


Figure 3. Firmware Board/Programming Card Circuits

decoder while the write enable signal is derived directly from the system bus. When these two signals are active, the timer (a one-shot) is triggered to time the 10ms (2816) or 50ms (2815) programming pulse. Concurrently, a busy signal is returned to the microprocessor bus for the duration of the write pulse. The write pulse from the timer is fed to the wave shaper which generates the 21V programming pulse needed to write an E^2PROM byte. Data and address are provided by the normal TTL bus signals.

When a small firmware error is found (a few lines of code) while debugging a prototype, the solution, in the form of patch code, can be entered directly via a front panel keyboard by the test engineer. This new code replaces the original code in a portion of the E^2PROM memory.

When several patches are made during the course of testing, or at a periodic interval, the latest revision level of code would be recompiled in its entirety on an off-line basis. It would then be loaded into the system. The recompiled code condenses all changes and makes sure that all printouts are up to date. The complete process of firmware revision described here maximizes the time which is spent actually checking out the prototype.

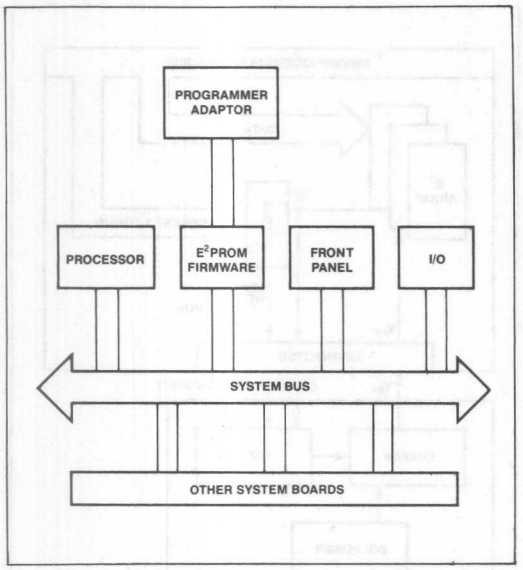


Figure 4. Prototype System Implementation

Usually, production of a new product begins before the prototypes are fully debugged and tested. Because these first production models may require final changes before leaving the factory, it is advantageous to use E^2PROMs in these models. This allows firmware modifications to take place on the production line using the programming daughter board, during the period when the firmware is not entirely finalized. These production units will not need to be dismantled as final firmware changes are made. When the system is completely checked out, the daughter board containing the E^2PROM programming circuitry can be removed from the production cycle. EPROMs or ROMs can then be plugged directly into the sockets which originally used E^2PROMs. At this stage, production will probably be fully under way.

Several advantages are derived from implementation of a firmware memory system in this way. Foremost is the compatibility of the production board with the prototype. This precludes the necessity of redesigning the memory board twice. A device swap from E^2 to ROM is all that is required when system firmware is finalized.

Since E^2 is programmed in-circuit, no extra inventory of PROM devices are required for different revision levels of code. The time it takes to replace ROM type devices and the problems of bent or broken pins, worn sockets, and worn connectors are eliminated.

Another advantage is the way all programming circuit costs for the E^2PROMs are distributed over all firmware memory boards. Often, several prototypes are being developed at once. The same programming module can be plugged in on all these systems. Also, the power supply for the 21V V_{PP} pulse on the E^2PROMs does not have to be resident in the final systems. For prototyping, power can be supplied by an external source or converted from 5V on the programming daughter board, eliminating the cost of an extra power supply from production models.

A final advantage is the cost savings on ROMs such a system can derive when a product needs a firmware update. No parts need changing if E^2 is the firmware storage memory!

October 1981

E^2PROMs as an Error Accumulation Medium

Lawrence Palley
Special Products Division

© INTEL CORPORATION, 1981.

INTRODUCTION

The servicing of electronic equipment is a business area currently receiving much greater attention than it has in the past. Companies are now examining a product's life in the field with increasing concern. This a result of the dramatic cost increase associated with servicing a product at a remote site (up to 100% of the system's original cost). To combat these cost increases, companies are looking for more ways to build serviceability into their products before they leave the factory. They are looking to new technologies for the answers.

A recent technological innovation in semiconductor memory has been the development of the Electrically Eraseable PROM by Intel. Its availability as a non-volatile memory alternative opens up a broad spectrum of service cost-reducing applications in MPU-based equipment. These applications include remote firmware updates, diagnostic storage, signature storage, and the subject of this note—error logging.

This brief will discuss some of the problems associated with system maintenance and the types of errors which can make diagnosis a costly venture. It will then suggest methods for applications of E^2PROMs as an error accumulation medium to aid in the solution of these problems. Finally, the benefits derived by using an E^2PROM solution are evaluated.

FAILURE ANAYSIS DIFFICULTIES

A troublesome area for engineers is the diagnosis and repair of problems indicated by intermittent errors. These problems could be related to changes in the environment, intermittent device failure, or infrequently used timing path constraints. They are particularly hard to detect because they occur during the realtime operation of the system.

Corrections for these problems are handled in a variety of ways depending on the severity. If the failures result in the shutdown of the machine, a customer engineer would be sent out immediately to diagnose the problem. Extensive diagnostics could be exercised and repairs result. On the other end of the severity spectrum, an error can be recognized by the user and ignored because it does not significantly affect his output. Because no formal record is kept, a serviceperson who is asked to fix the problem on a regular maintenance call, can only try and reenact the error condition. Many times this is not possible and several calls could be required to fix the problem. Even after these calls, the solution can be elusive and it may be required to do a total system swap. These two cases serve to show that whatever the error severity, a record of the errors could help a serviceperson quickly identify the problem.

E^2 APPLICATION AS AN ERROR LOG

There are two primary applications of E^2PROM as an error logging medium. The first involves using the E^2 as a general system log where errors occurring in the entire system are stored. This application is useful to log randomly and unexpected system problems. The second is application in a dedicated error logging, and possibly correcting, system.

The first class of E^2PROM error logging applications is depicted in Figure 1. A standard E^2PROM array is interfaced to the microprocessor bus. An additional auxiliary register is necessary to keep track of the address identifying the next byte of E^2PROM memory to be written in the event of an error. This register serves as an address pointer and may be implemented as a RAM location or an actual piece of hardware. The contents of this register would be saved in the last location of the E^2PROM array upon power down so that its information would not be lost.

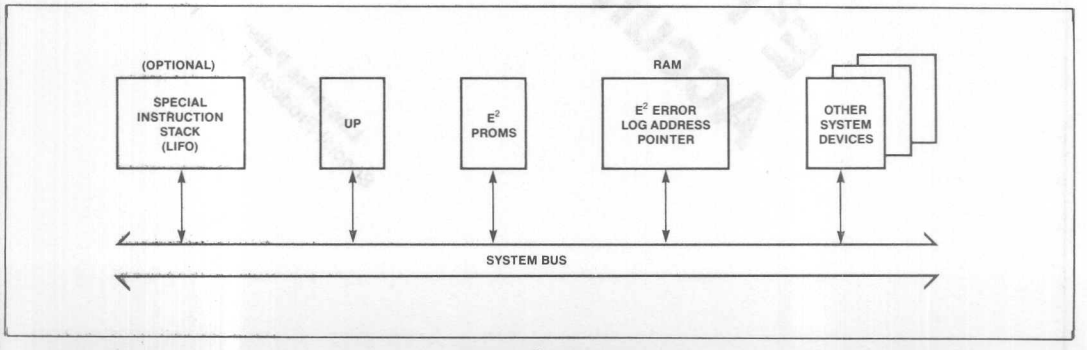


Figure 1. Typical E^2PROM Error Log

The stack is optional hardware, but can be very useful in diagnosing random timing path or noise errors. At the start of every instruction, the last instruction would be automatically pushed onto the LIFO stack. In this way, a record of the previous instructions is always at hand.

The error logging process would begin with error detection through normal software and hardware flags. Upon detection of the error, the trap or interrupt routine to handle and restart the processor would be entered. At the start of the routine, the next addressed E^2PROM byte is used to store system parameters. Instructions would be executed to store these vital system parameters including flags, status registers and the state of various CPU internal registers. A more sophisticated system would then be able to pop the stack (which was disabled at the start of the trap routine) and also store the code for previously executed instructions in the E^2PROMs. These instructions provide a system history. The final step in the routine would be to restore the E^2PROM address pointer register to reflect the next addressable E^2PROM byte for the next log entry.

Since the E^2PROM array is constrained by size, the situation of filling the entire array must be addressed. To prepare for this event the processor would, as part of the auxiliary register update routine, test the register to ensure that the next address is within E^2 bounds (Figure 2). If this is not the case, the processor would notify the operator that the array is full and/or dump the entire E^2PROM contents to a printer. The processor would

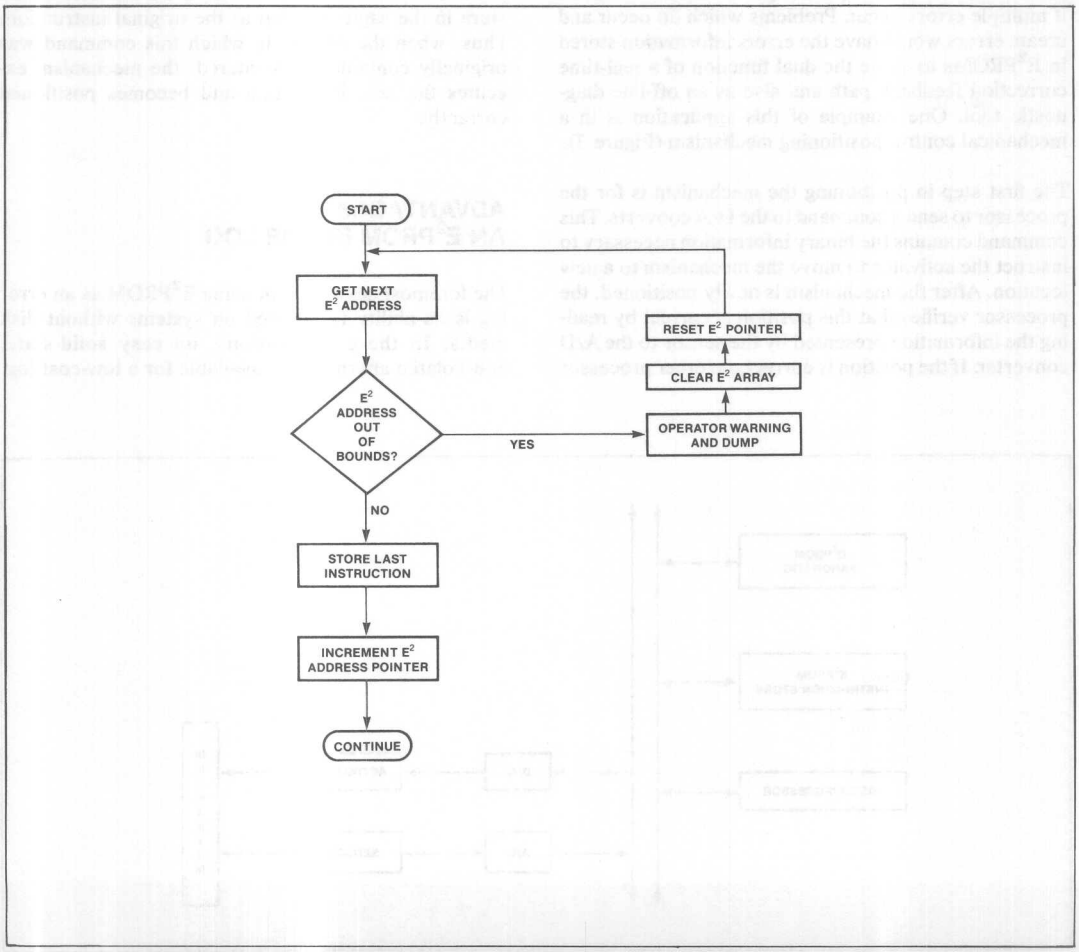


Figure 2. E^2 Address Pointer Update Routine

then, using the block erase features of Intel's E^2PROMs, erase the entire array and reset the address pointer. If it is suspected that the E^2 array will quickly refill because of a hard failure, the operator would be able to disable the error logging feature so as not to overrun the E^2PROM write capacity.

A repair person would be called in to fix faults. The error logs resulting in E^2PROM could be quickly analyzed by the service engineer to determine system problems. Maintenance on faulty modules or to implement Engineering Change Orders to correct hard or soft errors would then be executed.

A second class of E^2PROM applications are in those systems requiring dedicated error logging. In these applications, continuous system operation is desired even if multiple errors occur. Problems which do occur and create errors would have the errors information stored in E^2PROMs to serve the dual function of a real-time correction feedback path and also as an off-line diagnostic tool. One example of this application is in a mechanical control positioning mechanism (Figure 3).

The first step in positioning the mechanism is for the processor to send a command to the D/A converts. This command contains the binary information necessary to instruct the activator to move the mechanism to a new location. After the mechanism is newly positioned, the processor verifies that this position is correct by reading the information presented by the sensor to the A/D converter. If the position is correct, no other processor action is required. When the position is incorrect, the processor stores the instruction and the resulting positional information in a part of E^2PROM used as an error log. This log would be used at a later time, off-line, to aid in the diagnosis of the problem causing the incorrect position. After logging the fault, the instruction could be retried to make sure that the problem is not random. If, after several retries, the instruction continues to position the mechanism incorrectly, then corrective action needs to be taken. The fault is indeed one that does not change over time.

To provide corrective instructions the processor would execute a program intended to discover the instruction which will position the mechanism correctly. After this instruction is found (possibly through successive approximation), it is stored in the E^2PROM instruction store in the same location as the original instruction. Thus, when the routine in which this command was originally contained is reentered, the mechanism executes the new instruction and becomes positioned correctly.

ADVANTAGES OF AN E^2PROM ERROR LOG

The foremost advantage of using E^2PROM as an error log is its ability to be used on systems without disk media. In these applications, no easy solid-state, non-volatile alternative is available for a low-cost log.

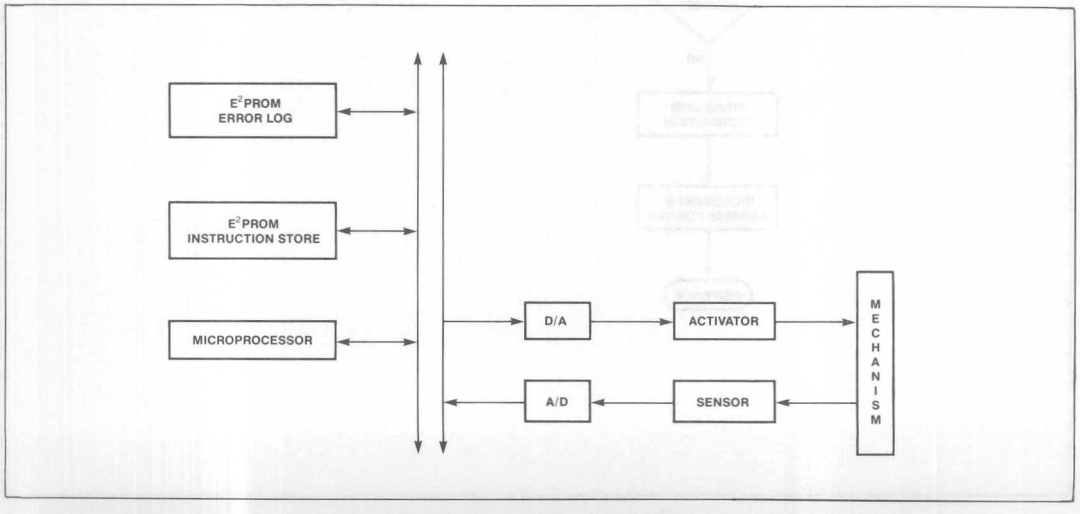


Figure 3. Mechanical Control Mechanism

An error log would substantially reduce the cost to service the equipment by significantly reducing the mean time to repair. In fact, the use of error logs can allow the user to diagnose and repair his own system—completely eliminating the service call.

Another significant advantage of E^2PROM is its solid-state nature when used in systems which do have rotating media. Since these devices are mechanical, they are the most prone to failure. By implementing an error log on E^2PROM, the log is accessible even after a head crash or some other disk failure. This reliability makes the capability for remote or local diagnosis an even greater certainty.

Use of E^2PROM in a dedicated environment, provides much needed logging flexibility while also providing fault diagnostic capability. Mechanisms located in remote or harsh environments which are not easily accessed by service engineers, can operate with a much higher degree of fault tolerance. Better reliability for continuous operation is the direct result.

Much reduced servicing costs plus much increased reliability are two key results manufacturers look for when designing new systems. E^2PROMs used as an error logging medium provide a flexible, easy-to-implement, and reliable alternative toward achieving these goals.

October 1981

Programmable Controllers and Data Loggers

Dennis Molnar
Special Products Division

© INTEL CORPORATION, 1981.

INTEL'S E²PROMS CAN BENEFIT BOTH THE MANUFACTURER AND THE USER

The Industrial setting is finding Programmable Controllers and Data Loggers to be growing in sales as the cost of electronic hardware decreases, and as processes become more complex. All too often, the Industrial Process is restrained, not by its speed and dependency limitations, but by that of its supporting devices. As the cost of electronics hardware goes down, the cost of process downtime is going up because of the increased cost of labor, materials and opportunity.

This brief discusses the typical system architecture of Programmable Controllers and Data Loggers, then examines the current use of battery-backed RAM in the systems, and notes benefits received from using Intel E²PROMs as an alternative to the battery-backed RAM.

Typical system architecture centers around a 16-bit microprocessor with a keyboard, display, tape drive, I/O and memory. Since the system operation consists of sequentially scanning the inputs, processor operation consists of retrieving an instruction from the user memory, executing it via the operating program, using the scratchpad, performing any output required, and repeating the cycle. Should the user program need to be loaded into user memory, it can be done by keyboard, tape load or a remote processor. See figures 1 & 2.

The battery-backed up RAM is used to store the user-specified program. It must be non-volatile as the user does not want the inconvenience of reloading it (by tape or keyboard) every time the power is interrupted. Should the process be of a batch nature, the microprocessor scratchpad and data table also need non-volatility so that if the process is interrupted by a power failure, it can be started again without initializing the process. In Data Loggers, the clock is battery backed up in order to provide correct program function on startup per time of day.

The RAM and battery back-up of the user program does serve its purpose but is susceptible to the rigors of the harsh industrial environment. Occasionally problems are caused for the user when the batteries or power sense circuitry fail to do their work. The inherent problem of batteries is that their lifetime of 1 to 4 years (depending on use) is not as long as the 10-year-plus life of the machine whose memory they back up. They must therefore be checked and replaced during the lifetime of the machine thus incurring high costs and paperwork for the user. The major concerns of industry users are twofold. First is that the batteries sometimes fail during a shut down. Second is that the power sensing circuitry can be affected by transients. Both of these can cause loss of the user program which hurts the user as it causes costly process downtime. The user does not blame all the problems of user memory loss on the reliability of the batteries and sensing circuitry. Often it is an operator who carelessly allows condensation, fluids or temperature extremes to be exposed to the

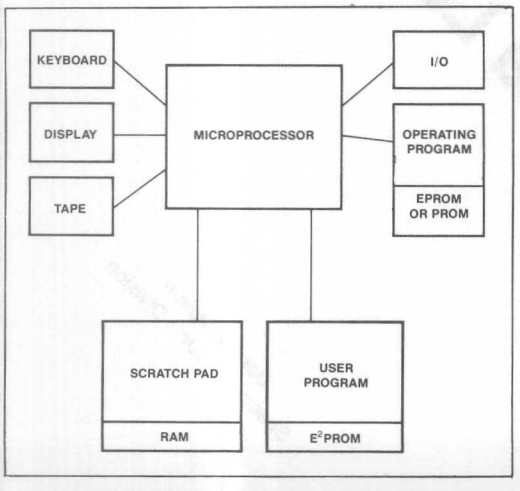


Figure 1. Programmable Control System Architecture

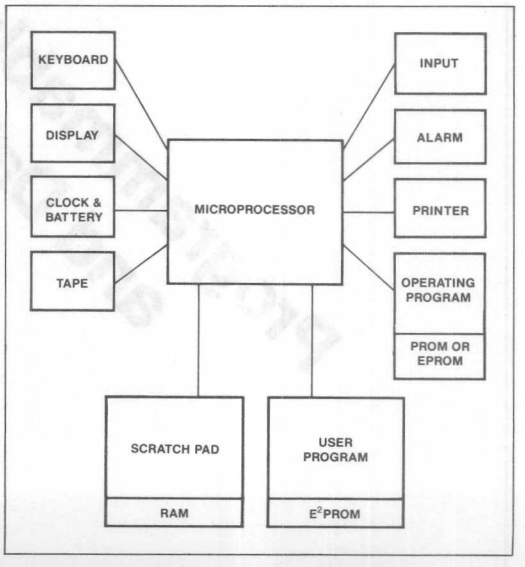


Figure 2. Data Logger System Architecture

Controller or Logger. All these factors can cause user memory loss through short circuits or through battery failure due to sensitivity to temperatures other than at room.

The costs of process downtime to the user can be from hundreds to thousands of dollars per hour. The user accepts that power failures are beyond his control but process downtime after a failure is controllable. If a user has to reload the program due to battery failure, insufficient charging, or sensing circuitry failure, the actual loading can take 10 minutes to many hours. This depends on how many machines are being used and and whether the user has already had experience (and downtime) and therefore knows the immediate solution from a previous repair. The Intel E^2PROMs offer the solution to drawbacks of battery-backed up user memory as they are *truly non-volatile* and offer the fast read access speed required by the microprocessor.

The benefits to the user are as follows:

—No battery failure during a shutdown necessitating a reload upon startup and production loss. Production savings using E^2 can be up to many thousands of dollars.

—No loss of user program due to power transients or operator negligence. Costly downtime can be avoided.

—Possible elimination of tape as a secondary storage medium saving an initial investment of hundreds of dollars.

—Elimination of battery costs and servicing costs for user program retention. This can be in the order of $10 or more each year depending on battery type, service method, and shut downs per year.

—Increased confidence in machine reliability.

The benefits to the manufacturer lie in the fact that over the life of the machine, costly customer production downtime will be avoided and therefore the customer will have a higher level of confidence in the machine's memory retention. This translates directly into a higher sales price for the manufacturer. Intel's E^2PROMs are a value adder to the manufacturer's products. They enhance product features, increase product value, reduce customer maintenance, reduce manufacturer's warranty costs and offer *true* non-volatile memory.

E^2PROM Applications

4

intel

APPLICATION NOTE

AP-100

June 1981

Reliability Aspects of a Floating Gate E^2PROM

Bruce Euzent, Nick Boruta,
Jimmy Lee and Ching Jenq
Intel Corporation

Based on presentation at 1981 International Reliability Physics Symposium, Orlando, Florida, April 7, 1981.

© Intel Corporation, 1981.

INTRODUCTION

Electrically Erasable Programmable Read Only Memories (E^2PROMs) that can be electrically erased and written one byte at a time are new components being used in computer systems. The E^2PROM is particularly attractive in applications requiring field update of program store memory or non-volatile data capture. It is only recently that E^2PROMs which operate via Fowler-Nordheim tunneling to a floating polysilicon gate have become available. The E^2PROM has the data retention requirements of earlier generations of PROMs, but also must maintain its field-programmable characteristics over its device life.

In this paper we shall first review the basic operation of the Intel® 2816 E^2PROM cell. Intrinsic failure mechanisms which limit the applications of E^2PROMs will be examined, and then defect mechanisms will be discussed. Finally lifetest data will be presented to predict operating failure rates.

Device Operation

The Intel 2816 uses the FLOTOX structure, which has been discussed in detail in previous literature[1]. Basically, it utilizes an oxide of less than 200Å thick between the floating polysilicon gate and the N+ region as shown in Figure 1.

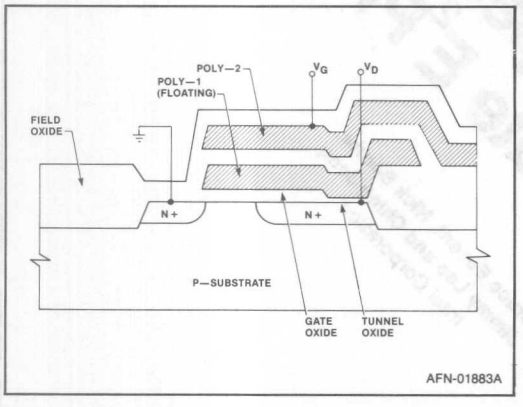


Figure 1. FLOTOX Device Structure Cross Section

Both erase and write are accomplished by tunneling the electrons through thin oxide using the Fowler-Nordheim mechanism[2]. The I-V characteristic of Fowler-Nordheim tunneling is shown in Figure 2, where the current is approximately exponentially dependent on the electric field applied to the oxide.

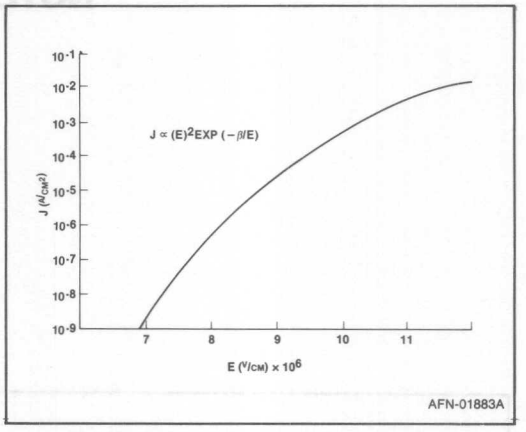


Figure 2. Fowler-Nordheim Tunneling I-V Characteristic

During the erase operation, approximately 20V is applied to the top gate of each cell in the byte while the drain is kept at ground potential. The electrical field in the thin oxide region is directed from the floating gate to the N+ region such that electrons tunnel through the oxide and are stored on the floating gate. This shifts the cell threshold in the positive direction causing the cell to shut off current flow and present a logical "1" at its output (as seen in Figure 3a).

On the other hand, when the cell is written to logic "0", the top gate is pulled down to ground potential and a high voltage is applied to the drain (with the source end floating). Electrons are depleted from the floating gate

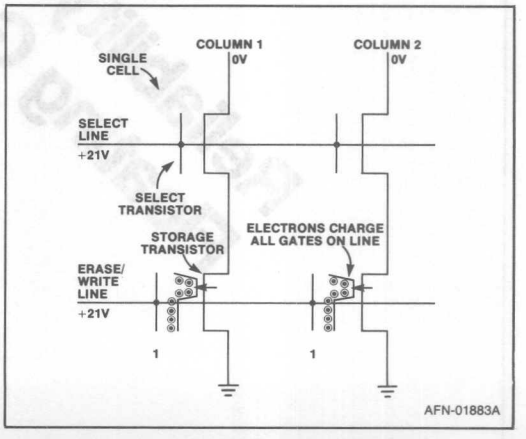


Figure 3a. Schematic of Memory Cell Operation During Erase

as seen in Figure 3b, and the cell is left with a negative threshold. Since the interpoly oxide is much thicker than the "tunnel oxide" and the electric field across the interpoly oxide is much smaller, the erase and write operations are predominantly controlled by the thin oxide region.

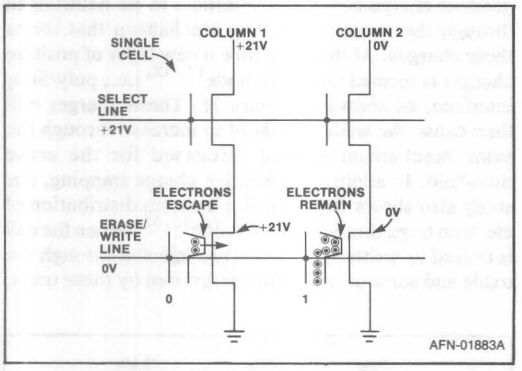


Figure 3b. Schematic of Memory Cell Operation During Write

Read Retention

The floating gate structure is known for its excellent charge retention properties. The reliability of this structure in the case of the EPROM device has been reported before[3]. The only remaining concern of the data retentivity of the 2816 is possible charge gain or loss through the tunnel oxide due to Fowler-Nordheim tunneling. The maximum electric field is built up across the tunnel oxide for a written cell, one that has a net positive charge on the floating gate. In this state the positive top gate voltage creates an electric field which adds to the field created by the positive charge on the floating gate, and there exists the probability that electrons may tunnel to the floating gate and shift the cell threshold. The band diagram of this condition is shown in Figure 4. However, the amount of current which may pass through the thin oxide during read or deselect is kept low by biasing the top gate of the memory cell at an internally generated voltage less than V_{CC}. The effect on the threshold shift of the cell can only be observed after long-term stress. Under this condition, the accelerated voltage test can be very useful.

If we assume Fowler-Nordheim tunneling is the predominant mechanism governing the movement of electrons, the threshold shift of the cell will be dependent solely on the voltage between the top gate and the N+ region. This has been proven to be true in both simulations and experiments, where we found that

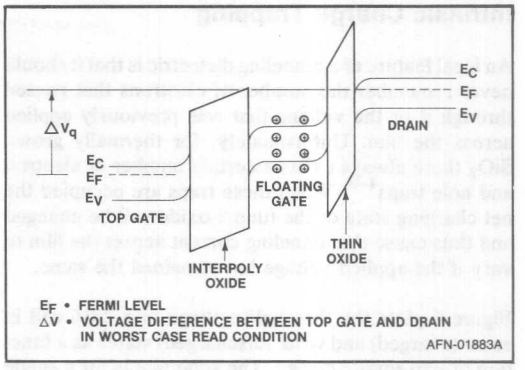


Figure 4. Band Diagram During Read of Written Cell

there is a one-to-one relationship between the V_T and the stress voltage. In other words, we can stress the device by applying a higher voltage to the top gate such that the change of the threshold voltage can be measured. The data then will be used to predict the same characteristics at the much lower normal read voltage. In Figure 5, the aforementioned simulation and experimental data are shown. The cell was biased at a voltage 4V higher than the normal read condition and the threshold voltage of the cell was monitored over a period of a week. A simulation was also generated to compare with the observed threshold shift and to demonstrate the technique we use to predict whether the data retention of the cell is accurate. As can be seen in the Figure 4, even under the accelerated voltage test the cell V_T still will not cross above the sense level after more than 10 years. Similar data has also been taken by writing the cell to a more negative initial threshold. In this case, the shift of the threshold can be observed at a stress of normal read voltage. Clearly, a 1V/1V relationship holds and an extrapolation can be made that the correct data will be retained for more than 10 years of continuous read.

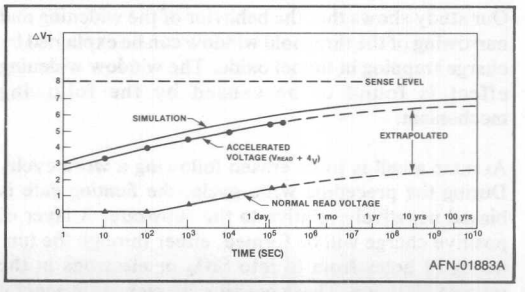


Figure 5. Single Cell Threshold Voltage Shift vs. Log Time During Read of a Written Cell

Intrinsic Charge Trapping

An ideal feature of a tunneling dielectric is that it should never remember the number of electrons that passed through it or the voltage that was previously applied across the film. Unfortunately, for thermally grown SiO_2 there always exists a certain number of electron and hole traps[4–9]. When these traps are occupied the net charging state of the tunnel oxide will be changed and thus cause the tunneling current across the film to vary if the applied voltage has remained the same.

Figure 6 plots the threshold voltage of a 2816 cell in erase (charged) and write (discharged) states as a function of erase/write cycles. The solid line is for a single cell, while the dashed line is for a typical 2816 array. It is seen that the threshold window, defined as the difference between the erase and write threshold, is increased in the first few E/W cycles and then saturates and remains almost constant until 10^4 cycles. From that point, the window begins to narrow gradually until around 10^6 cycles where the window is collapsed.

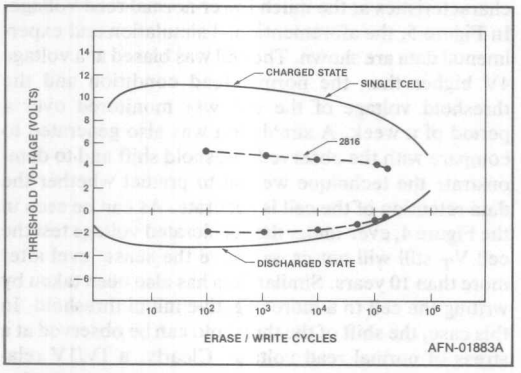


Figure 6. Typical Cell and Device Window vs. Log Cycles

Our study shows that the behavior of the widening and narrowing of the threshold window can be explained by charge trapping in tunnel oxide. The window widening effect is found to be caused by the following mechanism:

Assume a cell is to be erased following a write cycle. During the preceding write cycle, the floating gate is biased negatively relative to the substrate. A layer of positive charge will be formed, either through the tunneling of holes from Si into SiO_2 or electrons in the reverse direction. These positive charges are in general at 20–30Å away from the SiO_2/Si interface, as in Figure 7a. At the beginning of the erase step, the positive charges will cause an increase in electric field at the injection interface, i.e., SiO_2/Si interface, as shown in Figure 7b. This will in turn increase the tunneling current to the floating gate, where the amount of stored electrons is thus increased, causing the erase threshold to increase. During the erase cycle, however, the polarity of bias voltage across the tunnel oxide will cause the positive charge at SiO_2/Si interface to be neutralized through the reverse tunneling mechanism that forms these charges. At the same time a new layer of positive charges is formed near the anode[11, 12], i.e., poly/SiO_2 interface, as shown in Figure 7c. These charges will then cause the write threshold to increase through the same mechanism as that discussed for the erase threshold. In addition to positive charge trapping, our study also shows that there is a uniform distribution of electron traps throughout the oxide[11, 12]. When the cell is erased or written, electrons are injected through the oxide and some of them will be captured by these traps,

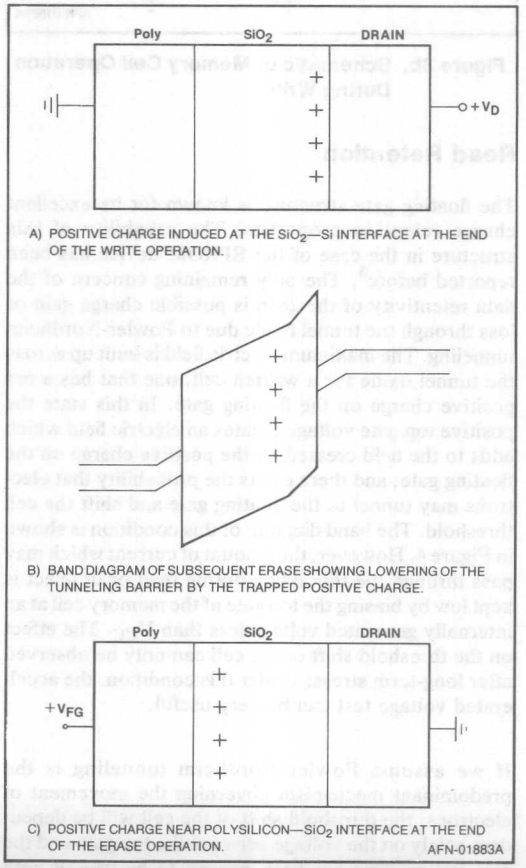


Figure 7. Threshold Window Widening

causing the build-up of negative charges in the oxide, as shown in Figure 8. The negative charges will reduce the electric field at the injection interface, thus decreasing the tunneling current and causing the threshold window to narrow. It has been found that the electron traps are not only preexisting in the oxide but also generated during the E/W cycles[9-12] because of the high field stress and the accompanying high current flow. The non-saturated build-up of negative charges, because of the continuous generation of electrons traps will finally cause the threshold window to collapse.

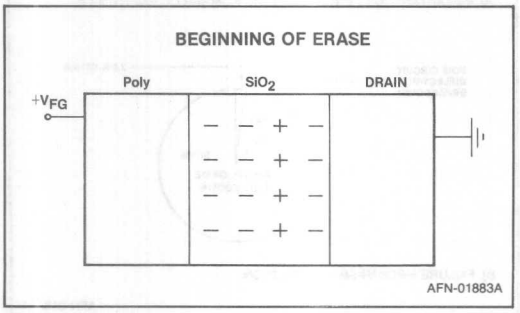


Figure 8. Negative Charges Trapped Uniformly Across Tunnel Oxide

Defect Charge Loss

EPROMs have been shown to have excellent data retention[3]. In this section we will discuss data retention studies that have been performed on the Intel 2816 E^2PROM. Since in E^2PROMs the number of Erase/Write cycles during the device lifetime is 3 to 4 orders of magnitude greater than in the EPROM, we will also need to address the effects of cycling on data retention.

As in the case of EPROMs the charge loss from the floating gate can be described as either intrinsic or defect-related. We will discuss the defect-related charge loss since the intrinsic charge loss on a typical device is identical to the EPROM and has been described before[3].

Devices exhibiting defect-related charge loss were erased to a logical "1" (electrons on floating gate) and stored at 250°C, 200°C and 150°C. The erase margins on the devices were monitored over various time intervals and the charge loss rate in volts per hour was determined. The results are shown in Figure 9. This data is normalized to 1 volt at 150°C. A best fit to the data shows an activation energy of .6 eV. This compares favorably to the defect-related charge loss observed in EPROMs.

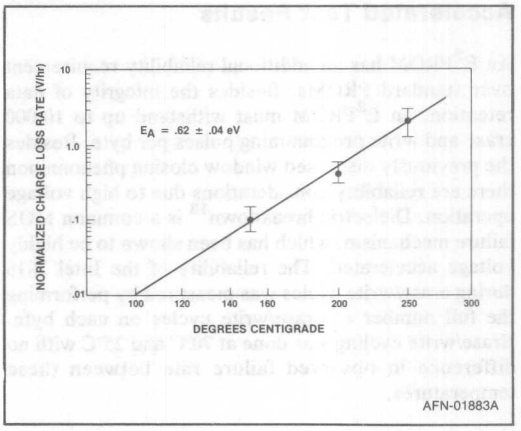


Figure 9. Plot of Defective Bit Charge Loss vs. Storage Temperature

Erase/write cycling effects on data retention were then studied by comparing 250°C retention before cycling to that after 10,000 cycles. Figure 10 shows a plot of the cumulative of % data retention failure during 500 hours 250°C retention bake. Data from the Intel 2716 EPROM is included as a comparison. From this data it is clear that cycling to 10,000 cycles has minimal if any effect on data retention. In addition, the retention failure rate closely resembles that of the Intel 2716 EPROM.

Since the defect charge loss failure mechanism is temperature activated it is simple to construct screens on a production basis for these types of failures similar to those used on EPROMs.

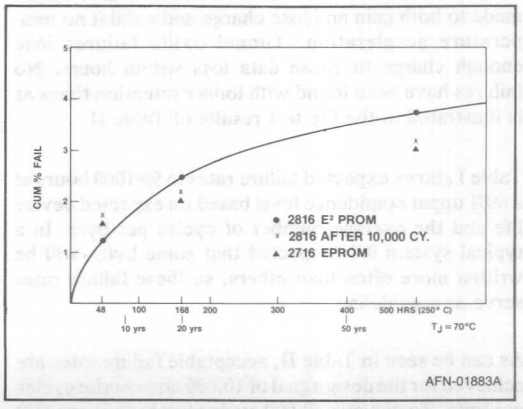


Figure 10. Intel 2816 Data Retention at 250°C, Percent Fail vs. Time

Accelerated Test Results

An E^2PROM has an additional reliability requirement over standard PROMs. Besides the integrity of data retention, an E^2PROM must withstand up to 10,000 erase and write programming pulses per byte. Besides the previously discussed window closing phenomenon there are reliability considerations due to high voltage operation. Dielectric breakdown[13] is a common MOS failure mechanism, which has been shown to be highly voltage accelerated. The reliability of the Intel 2816 during erase/write cycles was measured by performing the full number of erase/write cycles on each byte. Erase/write cycling was done at 70°C and 25°C with no difference in observed failure rate between these temperatures.

The results of erase/write cycling are shown in Figure 11A. The devices under test are completely tested after 2,000, 5,000 and 10,000 total cycles on each byte. The devices are programmed to several data patterns and tested to data sheet specifications. In addition, the devices are tested for high temperature data retention. As can be seen from Figure 11A, the failure rate per 1000 cycles decreases as a function of the number of cycles, which is typical for defect mechanisms such as dielectric breakdown[13].

Two major types of failures were found: Tunnel oxide breakdown and oxide breakdown in the row select circuitry. These failures were minimized by using standard screening techniques for oxide breakdown. Figure 11b shows the failure mode distribution found during erase/write cycling of 549 devices.

Tunnel oxide breakdown failures are cells which fail to erase or show conductive oxides. These failures can be made to both gain and lose charge and exhibit no temperature acceleration. Tunnel oxide failures lose enough charge to cause data loss within hours. No failures have been found with longer retention times as is illustrated in the life test results of Table II.

Table I shows expected failure rates in %/1000 hours at a 60% upper confidence level based on expected device life and the average number of cycles per byte. In a typical system it is expected that some bytes will be written more often than others, so these failure rates serve as a guideline.

As can be seen in Table II, acceptable failure rates are achieved for the design goal of 10,000 erase/write cycles per byte. To achieve 10,000 cycles per byte in ten (10) years, each byte must be altered approximately three times per day.

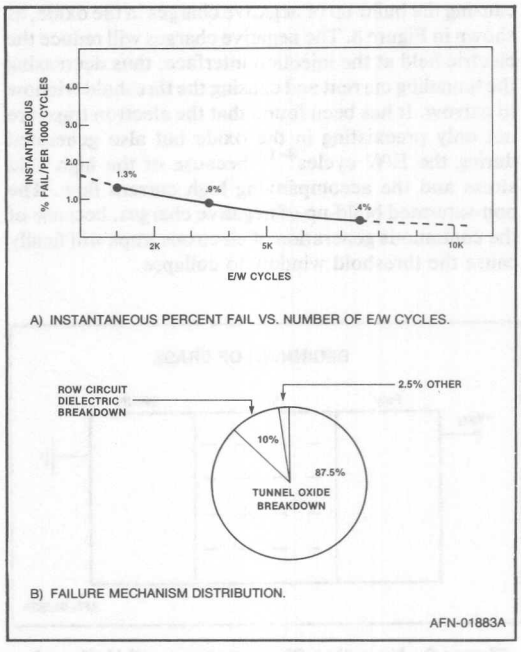


Figure 11. Erase/Write Cycling Results

Table I. Erase/Write Cycling Failure Rate (per 1000 hours at a 60% UCL)

Device Life	No. of Cycles		
	2000	5000	10,000
5 years	.065%	.11%	.17%
10 years	.032%	.054%	.087%
20 years	.016%	.032%	.043%

Table II. 125°C Lifetest Results

Cycles	48 Hrs	168 Hrs	500 Hrs	1000 Hrs	2000 Hrs
0	0/1422	1/1422[a]	1/443[b]	0/429	0/270
10,000	0/336	0/336	0/336	0/150	—
Total	0/1758	1/1758	1/779	0/579	0/270

Failure Analysis:
a) = Non-repeatable charge gain, contamination, lev.
b) = Input leakage, contamination, lev.

As a final verification of device reliability a standard high temperature lifetest at 125°C was performed on devices programmed with a checkerboard data pattern. The lifetest was performed on devices with no additional cycles and devices with 10,000 cycles on each

byte. As can be seen from the data in Table II standard MOS failure mechanisms were observed. This data is significant in that it shows no additional defect mechanisms related to data retention or erase/write cycling of the Intel 2816 E^2PROM.

Failure rate predictions are made in Table III at a 60% upper confidence level for both 55°C and 70°C operation. The .013%/1000 hrs. failure rate at 55°C shows good reliability comparable to other semiconductor memories.

Table III. Failure Rate Predictions at a 60% U.C.L.

125°C Device Hrs.	Activation Energy	Equivalent Hours		Lifetest Failures	Failure Rate % per 1000 Hrs	
		55°	70°		55°C	70°C
1.1×10^6	0.3 eV	6.8×10^6	4.5×10^6	0	.010	.020
1.1×10^6	0.6 eV	4.4×10^7	1.8×10^7	0	.002	.005
1.1×10^6	1.0 eV	5.3×10^8	1.2×10^8	2	.001	.003
				COMBINED	.013	.028

SUMMARY

This paper has discussed a number of E^2PROM failure mechanisms for both erase/write cycling and data retention. It has been shown that Fowler-Nordheim tunneling used for programming does not affect data retention. Erase/write cycling has been shown to degrade device margins by only a small amount and is easily guardbanded. Erase/write cycling does contribute to a significant portion of the observed failure rate due to oxide breakdown under high field operation. Defect-related charge loss has been shown to be similar to that observed in EPROMs. Finally, it has been shown that E^2PROMs can perform reliably in applications requiring up to 10,000 erase/write cycles per byte.

REFERENCES

1. W. S. Johnson, et al, "16-K EE-PROM Relies on Tunneling for Byte-Eraseable Program Storage," *Electronics*, February 28, 1980, p. 113–117.
2. R. Williams, *Phys. Rev.*, Vol. 140, p. 569, 1965.
3. R. E. Shiner, J. M. Caywood, B. L. Euzent, "Data Retention in EPROMs," 1980 Proceedings of the 18th Annual Reliability Physics Symposium, p. 238–243.
4. E. H. Nicollian, C. N. Berglund, P. F. Schmidt, I. M. Andrews, *J. Appl. Phys.*, Vol. 42, p. 5654, 1971.
5. M. H. Woods and R. Williams, *J. Appl. Phys.*, p 47, 1082, 1976.
6. W. C. Johnson, *IEEE Trans. Nucl. Sci.*, p. NS–22, 2144, 1975.
7. D. J. DiMaria, Proceeding of the International Topical Conference on the Physics of SiO_2 and its Interfaces, p. 160, 3/78.
8. E. Harari, *Appl. Phys. Letter*, p. 30, 601, 1977.
9. C. S. Jenq, "High Field Generation of Interface States and Electron Traps in MOS Capacitors," Ph.D. dissertation, Princeton Univ., 12/77.
10. C. S. Jenq, W. C. Johnson, "High Field Generation of Electron Traps in MOS Capacitors," presented in Semiconductor Interface Specialist Conference, 12/77.
11. C. S. Jenq, T. R. Rangarath, C. H. Huang, "Charge Trapping in Floating Gate Tunnel Oxide" presented in 1980 Non-Volatile Semiconductor Memory workshop to be published.
12. D. R. Young, "Electron Trapping in SiO_2", presented in 1980 Non-Volatile Semiconductor Memory workshop.
13. D. Crook, "Method of Determining Reliability Screen for Time Dependent Dielectric Breakdown," 1979 17th Annual Reliability Physics Symposium Proceedings, p. I-5.

April 1981

The 2816— Electrical Description

John F. Rizzo
Special Products Division
Applications Engineering

Flexibility, non-volatility, and a highly consistent system architecture — those attributes characterize the 2816 Electrically Erasable PROM. In this application note the electrical parameters that define the performance and operation of the device will be discussed. The concept of EPROM-like read architecture, encompassing high speed and 2-line control is detailed. In addition, the write/erase access needs some discussion as well. In the context of this discussion, the device performance, in its entirety, will be considered. In other application notes (Ap 102 and Ap 105), the system hardware and software architectural implications are discussed in detail.

INTRODUCTION

The 2816 is a 2K × 8 bit PROM that is electrically erasable. It's contents can be changed in the system without necessary removal from a board or cabinet. Along with this dramatic flexibility, the 2816 is non-volatile, just like the EPROM. The E^2 then benefits the user with EPROM-like data integrity and the additional capability to alter the memory data in-system. These two capabilities have never been possible with semiconductor memories. In addition to retaining data like the EPROM, the 2816 has very fast read access; data can be obtained from the device in less than 250 ns. This benefits system designers with high system performance to allow very competitive product entries.

The inherent flexibility that 2816 technology offers comes from the ability to alter single bytes of information. That is, just like a RAM, one byte of information can be erased and rewritten. Single-line editing of information is now possible. Direct register to memory transfer can occur without using additional and costly RAM buffer, which is unlike bulk erasable devices. In addition, if one wishes to erase the entire device at once, then a chip erase function is available. With this operation, all 2048 bytes of data can be returned to Logic 1 in 10 ms. The entire memory can be erased 300 thousand times faster than conventional EPROMs.

Because of the capability to write and erase data in-system, the 2816 architecture is designed to be very consistent. That is, the interface to the conventional microprocessor is simple and straight forward — unweildy and costly interface circuits are unnecessary. In the following paragraphs the read access, erase access, and write access modes will be discussed.

READ ACCESS MODE

The 2816 pinout, shown in Figure 1, is nearly identical to that of the 2716 EPROM. In the read mode, there are 3 groups of pins that are relevant: address, data, and control. The address input pins simply direct information within the device to be placed on the data output pins. When either of the control pins, $\overline{CE}$ or $\overline{OE}$ is logic "1", the data output pins are tri-stated. The combination of these control pins, called 2-line control, eliminates bus contention problems commonly encountered in microprocessor systems.

Chip enable is used as the primary device selection mechanism, and typically is obtained from address decoders. If chip enable alone is used to strobe data from the device to a common data base, then serious bus contention problems can result. Bus contention timing, shown in Figure 2, indicates why bus contention occurs. Basically, when one device on a common data bus is turned on, its outputs transition to either high or low levels. When it is deselected, there is a finite time delay before the output goes high impedance (this delay is a T_{DF} time which is specified in the data sheets).

Contention occurs, as shown, when one device is turning on while another is turning off. The timing overlap causes the data pins to be illegally driven from two sources. On any memory device with a single selection pin, system level bus contention can occur. Intel has pioneered the solution to bus contention through the use of the output enable pin. Output enable, as mentioned, simply strobes the output buffer. When output enable is connected to the microprocessor $\overline{RD}$ (read) line, contention is eliminated because no timing overlap can occur (as shown in Figure 3). Note that $\overline{CE}$ (derived from addresses) occurs far outside the $\overline{OE}$ signal — no overlap is thus possible. The two line control architecture of the 2816 therefore eliminates bus contention problems.

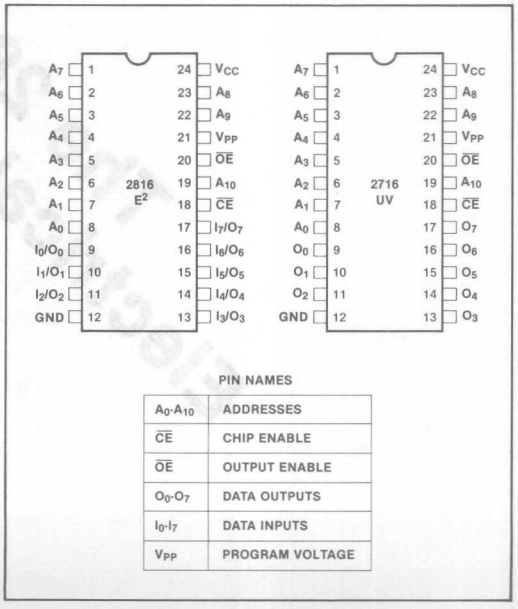


Figure 1. 2816 Pinout

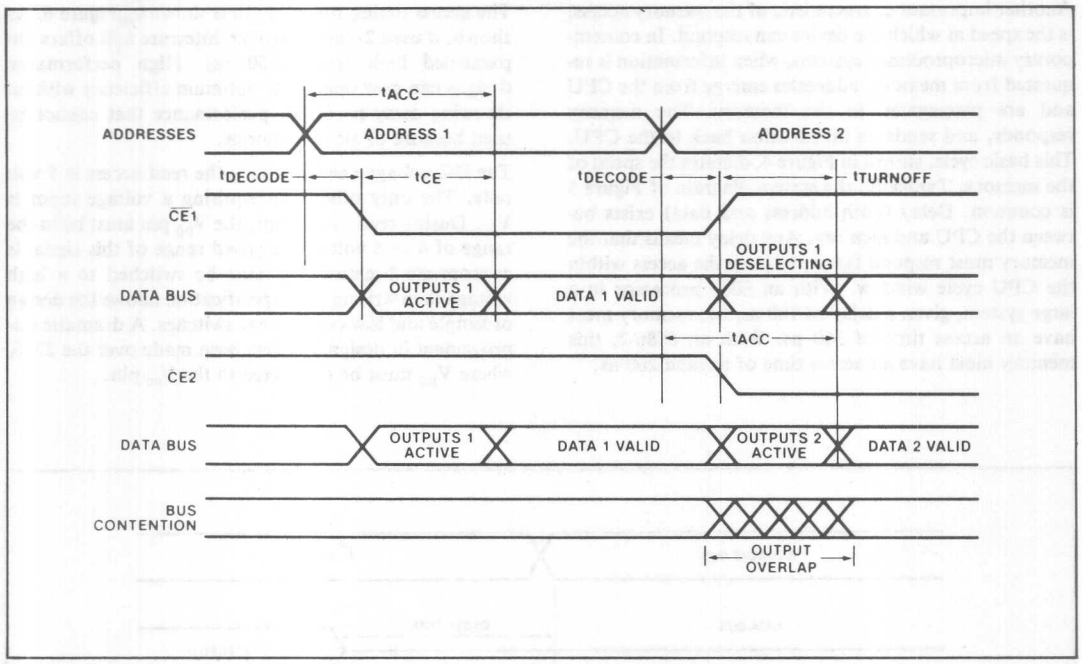


Figure 2. Single-Line Control and Bus Contention

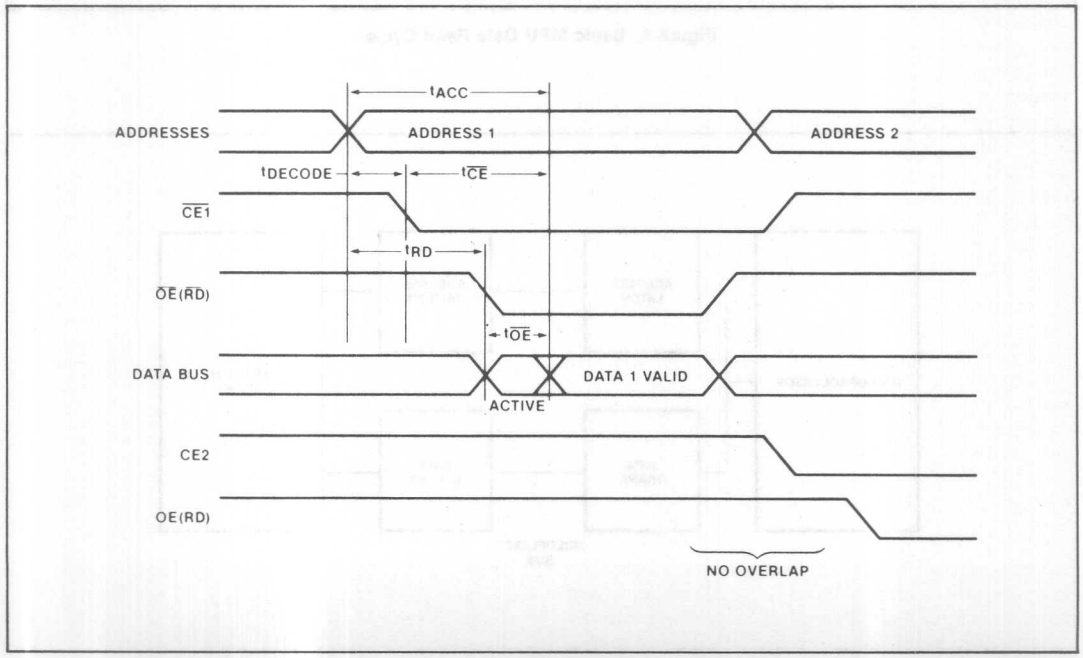


Figure 3. Two-Line Control Architecture

Another important characteristic of the memory access, is the speed at which the device can respond. In contemporary microprocessor systems, when information is requested from memory, addresses emerge from the CPU and are propagated to the memory. The memory responds, and sends its information back to the CPU. This basic cycle, shown in Figure 4, dictates the speed of the memory. Typically, the system diagram of Figure 5 is common. Delay (both address and data) exists between the CPU and memory. Any delay means that the memory must respond faster, to keep the access within the CPU cycle window. With an 8088 processor in a large system, given a delay of 100 ns, the memory must have an access time of 360 ns. With an 8086-2, this memory must have an access time of around 200 ns.

The access timing for the 2816 is shown in Figure 6. As shown, it used 2-line control architecture and offers unparalleled high speed (250 ns). High performance designs can now operate at optimum efficiency without throwing away processor performance that cannot be used because of slow memories.

The DC voltage needed during the read access is 5 volt only. The only other pin requiring a voltage input is V_{pp}. During read operations, the V_{pp} pin must be in the range of 4 to 6 volts. The broad range of this signal is appropriate because V_{pp} must be switched to a high voltage then writing. The specification allows the design of simple and low cost voltage switches. A dramatic improvement in design ease has been made over the 2716, where V_{pp} must be connected to the V_{cc} pin.

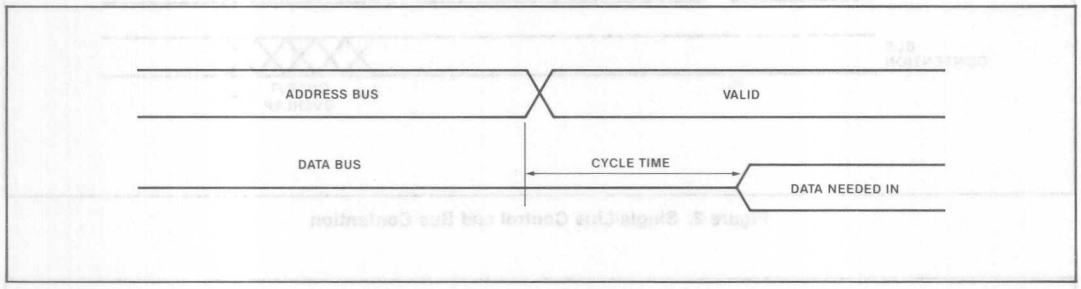


Figure 4. Basic MPU Data Read Cycle

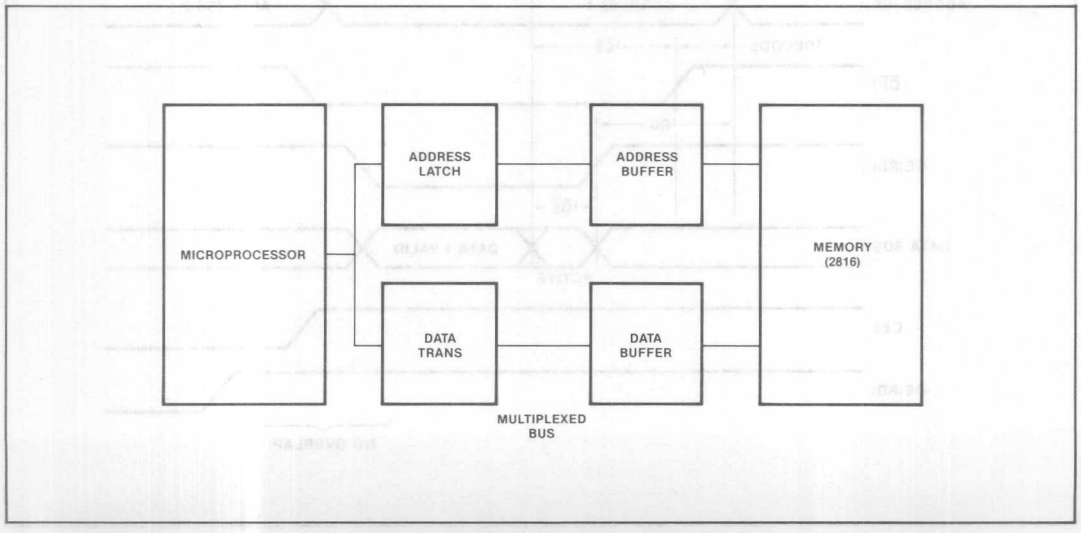


Figure 5. Common System Architecture

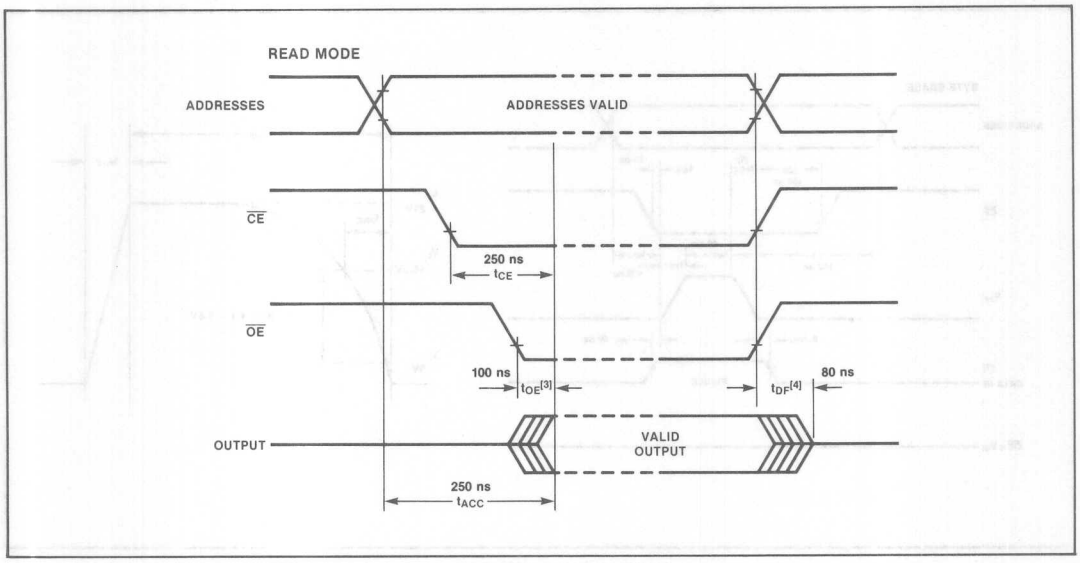


Figure 6. 2816 Read Access Timing

ERASE ACCESS MODE

The information stored in 2816 memory can be erased or changed through the application of simple electrical signals. A single, 10 ms, 21 volt pulse is all that is necessary to change any byte of information. The byte of data that needs to be altered must first be erased, then written.

The erase operation occurs automatically when certain information is presented to the 2816. In most cases, the byte must be erased prior to a data write. Whenever a bit within a byte must transition from a Logic 0 to 1, that byte must first be erased. Transitions from 1's to 0's can occur without an erase operation. Reasons behind the necessity for byte erase have been discussed in AR-118.

Mode selection for the 2816 is shown in Figure 7. The careful reader will note that the write and erase modes are basically identical with exception of the data input pins. When the input pins are all Logic Level "1", an automatic erase operation occurs. When a data pattern of ones and zeroes are presented, that data pattern is imbedded into the 2816 array. To accomplish byte erase the 2816 is selected by bringing $\overline{CE}$ to a logic Low. The address is provided to the device as well. To erase, a data input is set to "FF" Hex. The V_{pp} is then pulsed, through an exponential, to 21 volts. The timing diagram for this operation is shown in Figure 8. Note that there are set-up time requirements for address and V_{pp} to chip enable. At the completion of the write cycle, there are hold time requirements from V_{pp} as well. V_{pp} must rise through an exponential specified by an RC time constant, and be held for a minimum of 9 ms. V_{pp} can fall as quickly as possible, in fact, V_{pp} should be driven to 4 to 6 volts immediately to allow reading from the device, after a write. V_{pp} must rise slowly to 21 volts to allow low-level cell current flow to minimize cell voltage potentials. Simple circuitry is needed to provide this rise, and is explained in AP 102. During the entire erase cycle the output enable pin is kept at a VIH level. This makes much sense from a system compatibility standpoint since $\overline{OE}$ is an active low signal for read functions, and when high is inactive for erase/write functions.

In the erase mode $\overline{CE}$ is brought low. Microprocessor consistancy is preserved in this case as well because $\overline{CE}$ is derived from decoded addresses. The same address decoding circuitry — and nothing more — can be used to select the device in either READ or ERASE modes. This makes the system implementation very simple and straightforward.

PIN / MODE	$\overline{CE}$ (18)	$\overline{OE}$ (20)	V_{PP} (21)	INPUTS/ OUTPUTS
READ	V_{IL}	V_{IL}	+4 to +6	D_{OUT}
STANDBY	V_{IH}	DON'T CARE	+4 to +6	HIGH Z
BYTE ERASE	V_{IL}	V_{IH}	+21	$D_{IN} = V_{IH}$
BYTE WRITE	V_{IL}	V_{IH}	+21	D_{IN}
CHIP ERASE	V_{IL}	+9 to +15V	+21	[11] $D_{IN} = V_{IH}$
E/W INHIBIT	V_{IH}	DON'T CARE	DON'T CARE	HIGH Z

Figure 7. Mode Selection $V_{CC} = \pm 5V$

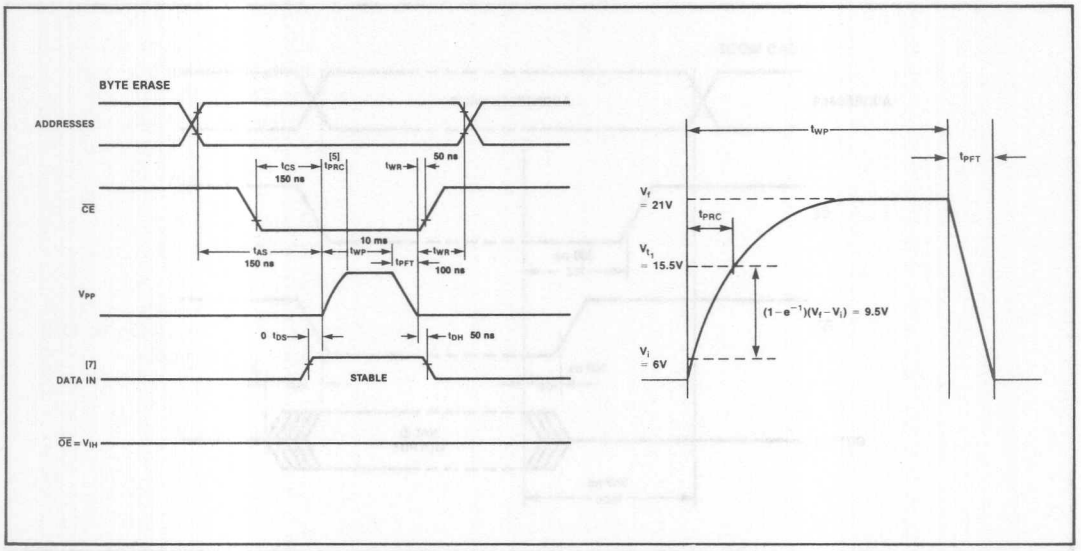


Figure 8. Byte Erase Timing

WRITE ACCESS

From the standpoint of functionality, the write access mode is identical to the erase mode. All setup times, hold times, voltage and timings are the same as used to erase the device. The only difference in operation is the data that is presented to the 2816. When a write is to occur, the data that is to be written is simply supplied to the device. The V_{pp} pin is pulsed exactly as before, all rise times and timings are consistent with the erase mode.

The timing diagrams for the write mode are shown in Figure 9. Also noted in that Figure are the actual device timing parameters.

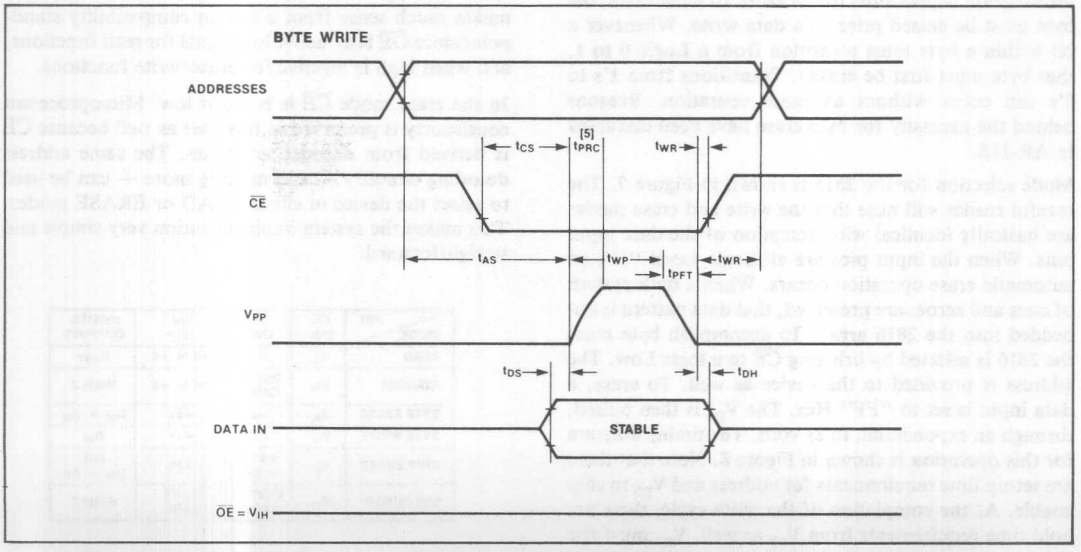


Figure 9. Byte Write Timing

In general, the 2816 has been designed to allow simple and straight forward mode selection and timing. In the erase/write mode, the control and functional pin designations reflect an in-system writable architecture. The design closely approximates RAM architecture to make system design easy.

The 2816 differs substantially from the 2716 EPROM in the write mode. The mode select tables for both devices are shown in Figure 10. In all cases, the 2816's functionality optimizes read and write operations above and beyond those inherent in the 2716 EPROM. All of the modes reflect a goal of simple designs in microprocessor systems.

IMPROVED

IMPROVED

MODE \ PIN	$\overline{CE}$ (18)	$\overline{OE}$ (20)	V_{PP} (21)	V_{CC} (24)	INPUTS/ OUTPUTS	
READ	V_{IL}	V_{IL}	+4 to +6	+5	D_{OUT}	2816
	V_{IL}	V_{IL}	+5	+5	D_{OUT}	2716
STANDBY	V_{IH}	DON'T CARE	+4 to +6	+5	HIGH Z	2816
	V_{IH}	DON'T CARE	+5	+5	HIGH Z	2716
BYTE ERASE	V_{IL}	V_{IH}	+21	+5	$D_{IN} = V_{IH}$	2816
	N/A	N/A	N/A	N/A	N/A	2716
BYTE WRITE (PROGRAM)	V_{IL}	V_{IH}	+21	+5	$D_{IN} = D_{IN}$	2816
	V_{IH}	V_{IL}	+25	+5	$D_{IN} = D_{IN}$	2716
E/W (PROGRAM) INHIBIT	V_{IH}	DON'T CARE	DON'T CARE	+5	HIGH Z	2816
	V_{IL}	V_{IH}	DON'T CARE	5	HIGH Z	2716

IMPROVED IMPROVED IMPROVED

Figure 10. 2716 Mode Selection

CHIP ERASE ACCESS

In order to erase all 2K bytes in 10 ms, special signalling is required. The output enable pin has been multiplexed for Chip Erase functions. To put the 2816 in that mode, $\overline{OE}$ is set in the range of 9 to 15 volts. Once engaged, the chip erase occurs by simply pulsing V_{pp} and $\overline{OE}$ in the same way as the write and erase modes. While a higher voltage is needed to perform chip erase, virtually no current flows into the $\overline{OE}$ pin. A standard 10 μA leakage current is specified over the full voltage range.

The timing diagrams and specifications for this mode are shown in Figure 11. The careful reader will notice that all of the signals (with the exception of $\overline{OE}$) are identical to the write/erase access modes.

DC VOLTAGE CONDITIONS

In the write and erase modes, the V_{PP} signal must be held within the 20 to 22 volts operating range. The 21 volt typical voltage is derived from Intel's patented HMOS-E processing. In the long term this will become a standard level for program voltages. If greater than the maximum of 22 volts is applied to the 2816, permanent and destructive device damage will result. If less than 20 volts is applied, then long term data retention is not guaranteed. The DC specification for the device is shown in Figure 12.

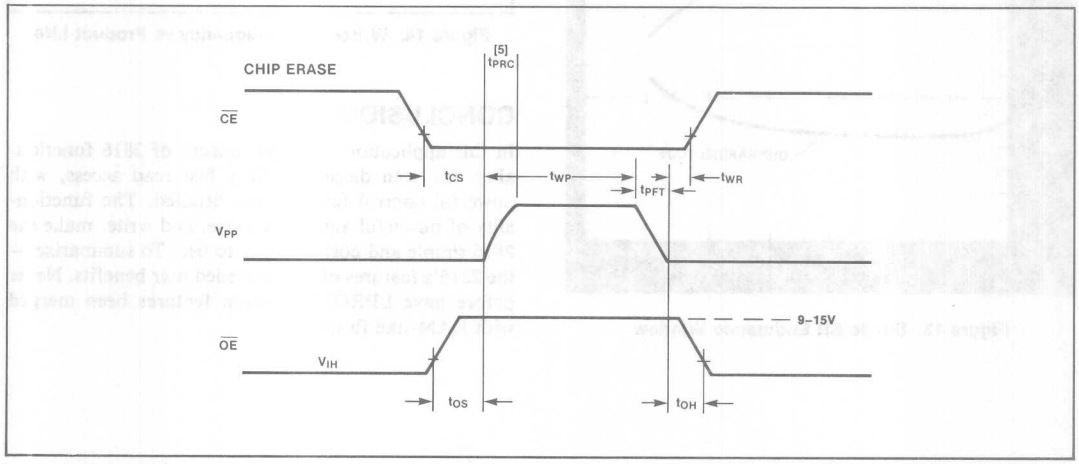


Figure 11. Chip Erase Timing

WRITE OPERATION

SYMBOL	PARAMETER	LIMITS			UNITS	CONDITIONS
		MIN	TYP	MAX		
V_{PP}	WRITE/ERASE VOLTAGE	20	21	22	V	
$I_{PP(W)}$	V_{PP} CURRENT (WRITE/ERASE)			15	mA	$\overline{CE} = V_{IL}$
V_{OE}	$\overline{OE}$ VOLTAGE (CHIP ERASE)	9		15	V	$I_{\overline{OE}} = 10\mu A$
$I_{PP(I)}$	V_{PP} CURRENT INHIBIT			5	mA	$V_{PP} = 21$, $\overline{CE} = VIH$

Figure 12. Write/Erase DC Parameters

ENDURANCE ISSUES

The 2816 has a characteristic ceiling on the number of erase/write cycles that can be endured. This ceiling exists because the cell threshold window changes (or closes) as the device is cycled.

Eventually, the device becomes permanently erased. Figure 13 shows how the single bit window changes.

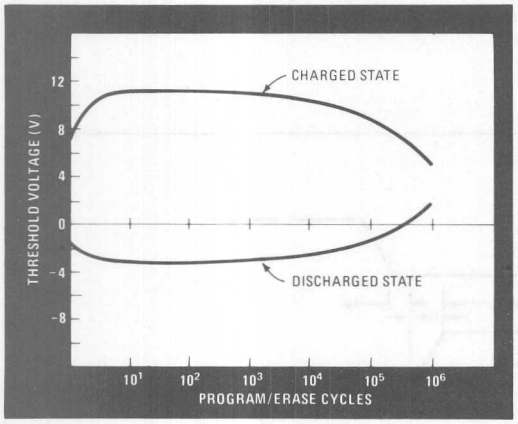


Figure 13. Single Bit Endurance Window

The E^2PROM from Intel is specified to handle 20,480,000 erase/write cycles per chip. Each byte can be cycled up to 10,000 times, and each byte operates independently of any other. Given a ten year machine life, each byte can be cycled up to 3 times per day. Figure 14 shows a graph relating product life and maximum write/erase frequency. In the majority of applications, less than 3,000 cycles are required.

This makes the 2816 an ideal device for those systems.

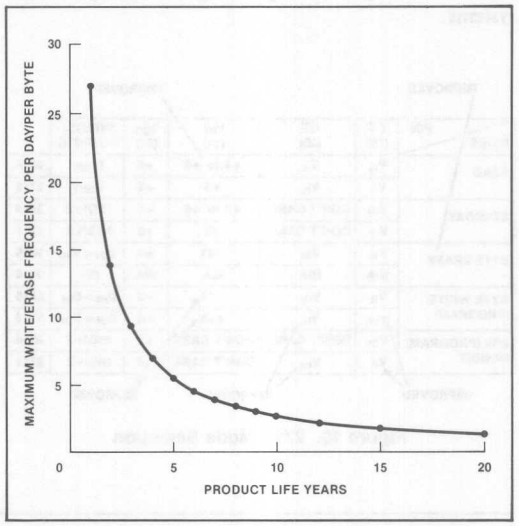


Figure 14. Write/Erase Frequency vs Product Life

CONCLUSION

In this application note the concept of 2816 functionality has been discussed. Very fast read access, with powerful control features was detailed. The functionality of powerful automatic erase, and write, make the 2816 simple and cost-effective to use. To summarize — the 2816's features offer unexcelled user benefits. Never before have EPROM retention features been merged with RAM-like flexibility.

intel

APPLICATION NOTE

AP-102

April 1981

2816 Microprocessor Interface Considerations

John F. Rizzo
Special Products Division
Applications Engineering

AFN-01913A

INTRODUCTION

E^2—Electrically Erasable, that's the key to the new 2816. The flexibility of RAM and the non-volatility of ROM have now been merged to form E^2. System designers can now benefit from in-circuit changes to non-volatile program and data storage. Microprocessor-based systems can be extended to a higher level of functionality and performance, while costs associated with software changes, maintenance and service can be dramatically reduced. A ROM with RAM-like flexibility—that's E^2.

This application note will discuss the concept of microprocessor interface to the 2816. Because E^2 encompasses both RAM and ROM, the interface concepts are unique. In this note, the control interface will be discussed specifically (four of which are detailed here). The concept of V_{PP} switching, and chip erase control circuits are also presented. Finally, using multiple 2816's in-system will be shown. In previous application notes (AP-101) the component characteristics were discussed. Here we will detail the interface of the component to the processor.

The specifications of the 2816 have been discussed in detail in AP-101. The most unique characteristic of the interface with the microprocessor is the concept of the write access. The read operation is fairly straightforward in that it does not depart from traditional EPROM concepts. The read operation is very fast, allowing compatibility with current and future microprocessors, benefiting the user with highest possible throughput and system performance. Because the write cycle time is not the same as read access, a unique situation exists for the system designer.

Because the 2816 requires a write time of approximately 10 milliseconds, there is an intrinsic timing difference between the microprocessor and the memory. If one applied the 10 millisecond write time to the write cycle time of the microprocessor, one could execute approximately 50 thousand write cycles in the duration of 10 milliseconds. Additional circuitry is required to properly interface these timing differences. There are several approaches for doing this, several of which will be discussed.

BUS INDEPENDENT TRANSFER

These approaches can be broken down into two general categories: bus dependent and bus independent. The bus independent concept allows the microprocessor to run at full speed while the 2816 write operation progresses. The microprocessor sends out a write operation just as usual, except that a control interface continues the 10ms write cycle independent of the CPU. The microprocessor is notified at some later time that the write operation is finished. This can occur either through interrupt service, or through an I/O polling operation. Thus, the microprocessor can run independently of the E^2 controller during the write time. Appropriately, it is "bus independent." Table 1 shows a partial list of appropriate applications using this controller type.

Table 1. Bus Independent Applications

CRT Terminal Control
Navigation Computers
Industrial Controllers
Telecommunications
Military Computers

BUS DEPENDENT TRANSFER

The other approach involves dedicating the microprocessor during the E^2 write cycle. In this case wait states are inserted into the memory cycle as the write is proceeding. The disadvantage of such an approach is that the microprocessor is inhibited from doing any other operation during the 10 millisecond write time.

In many applications, however, this can be a suitable solution to the 2816 control issue. An example is the case where information is transferred into the E^2 on system power up or power down. During the power sequencing times, one expects that the system would not be executing any other instructions, or in fact, doing anything other than servicing the E^2 device. In terms of hardware, this scheme would be implemented by controlling the microprocessor's ready or wait line while the write is occurring. This approach offers the advantage of being very simple to implement and does not require any software overhead in terms of interrupt service or I/O polling. Additionally, this scheme is acceptable in many applications where erase/write is only occasional. Such an interface is termed bus dependent. Table 2 provides an applications guide for this interface.

Table 2. Bus Dependent Applications

Program Storage
Look-up Tables
Remote Data Collection

We will show that the two distinct control applications dictate the amount of hardware required to interface the device to the microprocessor, as well as the efficiency at which the information transfer occurs. Above all, the individual application area for the E^2 will uniquely determine the kind of control circuitry that is required.

Based on these two distinct areas, we will discuss several different recommended interfaces that have been generated for use with the device. Though these controllers were designed to operate in an 8085/8088/8086 based system, they can be easily adapted to any kind of microprocessing environment.

INTERFACE OVERVIEW

There are five controllers at present, four of which are available for use with the 2816 Demonstration Unit. The Controller I is a small scale integration implementation which uses the microprocessor's ready line as a means of inserting wait states into the memory cycle. It is a very simple controller application; one that is dedicated to the microprocessor. For this controller, the microprocessor is inhibited from operating during the time that the 2816 is being written to. Figure 1 is a block diagram for this control interface.

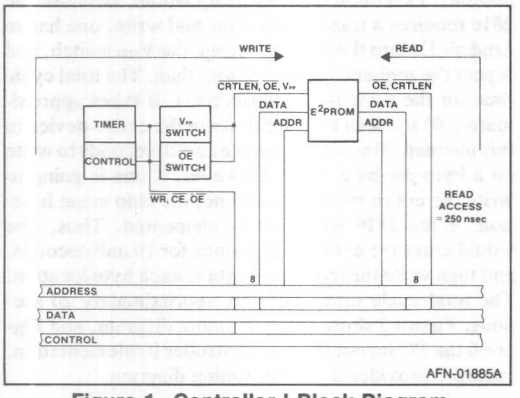


Figure 1. Controller I Block Diagram

The Controller II implementation is an interrupt driven interface, which requires little software overhead. In this case, the information is sent into the interface while the microprocessor simply strobes the write line as normal. The controller then handles all the necessary latching and generation of signals for the E^2 device. At the completion of the write cycle, the controller signals the microprocessor with a restart vector to interrupt service routines. The block diagram for Controller II is shown in Figure 2.

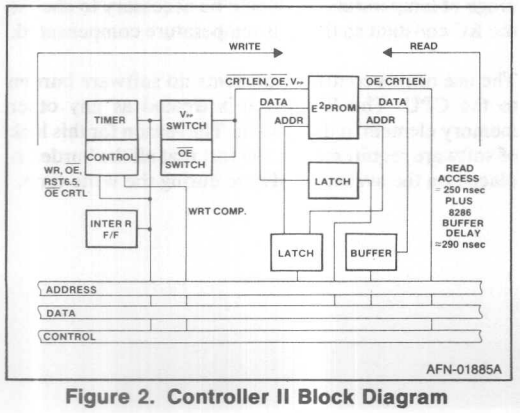


Figure 2. Controller II Block Diagram

The Controller III design is a more integrated version of II; it uses an Intel 8155 for controlling, latching, timing, and other functions. This controller, however, requires software in order to drive the 8155 and to set up the proper address/data lines to the 2816 during the write cycle. See Figure 3 for this block diagram.

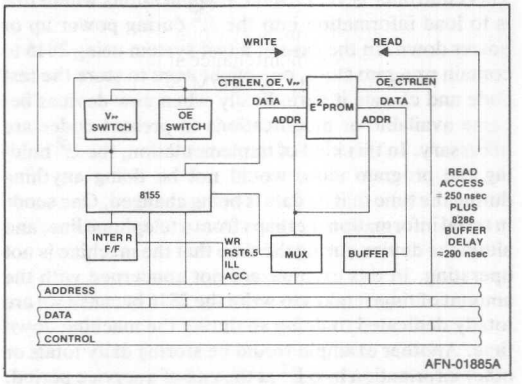


Figure 3. Controller III Block Diagram

The Controller IV implementation is a more highly integrated version of III; it uses an 8155 for writing and reading of the 2816. It also requires more software for the necessary initializations. A block diagram is given in Figure 4. Controllers I through III allow the 2816 to be read at very high speeds. Controller IV, however, requires long read times as reading occurs through the 8155 I/O port.

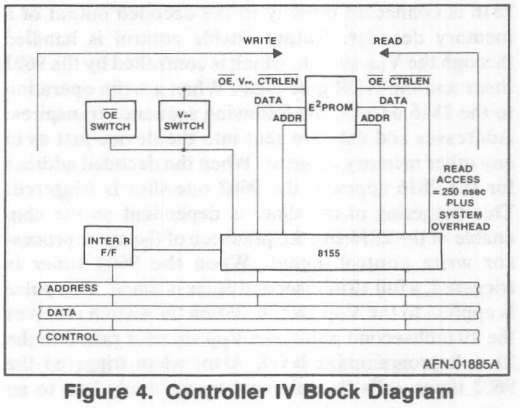


Figure 4. Controller IV Block Diagram

Controller V is an interface using a Bipolar PROM as a state machine. In this case there are two separate addresses for the E^2 device in the system; each of which corresponds to a different controller function. The first address corresponds to reading and writing of the E^2, the second address to chip erasure of the 2816. This controller is easily applied where a large memory space is available, as in a 16-bit microprocessing system.

CONTROLLER I DESCRIPTION

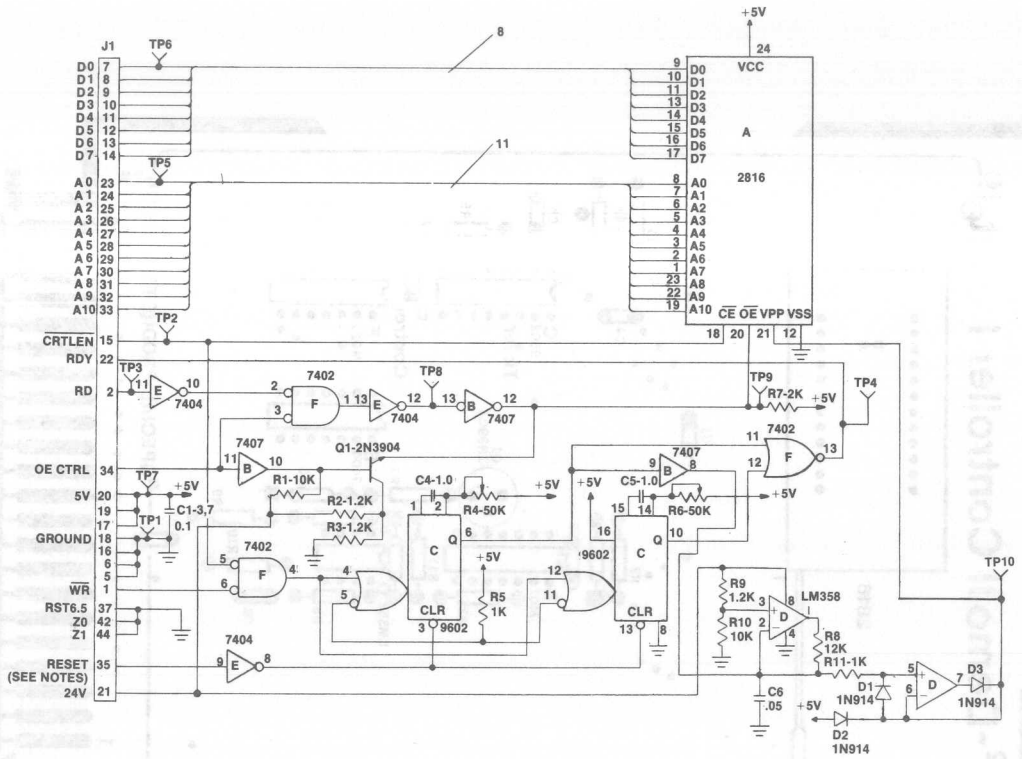
Examining the controller implementations in more detail, we find that the Controller I interface inhibits the microprocessor from operating during the write time. This controller is very useful in applications where one is to load information into the E^2 during power up or power down. In the case of a test system using 2816 to contain program store, one might want to store the test code and change it periodically when new devices become available or modifications to present codes are necessary. In this kind of implementation, the E^2 holding the program store would not be doing anything during the time that its data is being changed. One sends in serial information, perhaps from a telephone line, and alters the device during the time that the machine is not operating. In this case we are not concerned with the amount of time it takes to write the 2816 because we are totally dedicated to doing so during the machine down time. Another example would be storing daily totals or other information into E^2 at the end of a service period. In this case, when the machine is powered down it will automatically update the 2816 as a data memory. The amount of time it takes to do this is irrelevant because the machine is totally dedicated to the task during its shut down period.

The Controller I implementation discussed here uses three components in the system, shown in Figure 1. The 2816 address and data lines are connected directly onto the microprocessor bus. The chip enable line for the 2816 is connected directly to the decoded output of a memory decoder. Output enable control is handled through the V_{PP} switch, which is controlled by the 9602 timer and the NOR gate logic. When a write operation to the 2816 occurs, the following sequence transpires: Addresses and data are sent into the device just as in any other memory element. When the decoded address for the 2816 appears, the 9602 one-shot is triggered. This triggering of the timer is dependent on the chip enable of the 2816 and the presence of the microprocessor write control signal. When the 9602 timer is triggered, a full 10 millisecond pulse is timed. This pulse is applied to the V_{PP} switch. When the switch receives the 10 millisecond pulse, the V_{PP} signal is raised to the 21 volt programming level. Also, when triggered the 9602 timer pulls the microprocessor ready line to an inactive low level. This signals the microprocessor that the memory element is not ready to relinquish the data bus, or indeed requires a long write time.

The ready line inhibits the microprocessor from incrementing the program counter and causes the processor to provide stable signals to the 2816 during the 10ms pulse. At the completion of 10ms, the timer disengages the V_{PP} switch, stopping the write. It also pulls the microprocessor ready line to high level. When the ready line is pulled high, it indicates that the memory element has completed its cycle and that the microprocessor can continue execution as it normally would. Because the 2816 requires a transparent clear and write, one has to send all 1's into the device, engage the V_{PP} switch, and repeat the sequence for the proper data. The total cycle time for the write is 20 milliseconds. It takes approximately 40 seconds in order to write the entire device in this manner, 20 seconds to erase and 20 seconds to write on a byte-per-byte basis. However, if one is going to write the entire block at one time, the chip erase function of the 2816 would be implemented. Thus, one would erase the entire chip at once for 10 milliseconds, and then write the individual data at each byte location. The total cycle time would be approximately 20 seconds. Figure 5 shows the schematic diagram, and Figure 6 the PC layout for this controller implementation. Figure 7 provides a system timing diagram.

The components mentioned were chosen for Controller I more for convenience than for circuit design requirements. Conceivably, one could have other devices operating in the system to provide timing of the 10 millisecond pulse and switching of the V_{PP} signals. A programmable timer could exist within the microprocessing environment and could time out the 10 milliseconds more accurately than is possible with the 9602. One of the difficulties with the one-shot is the inherent variability of the RC time constant used to time 10 milliseconds. If the system is to operate over a wide range of temperatures, it would be necessary to choose the RC constant so that it is temperature compensated.

The use of this controller presents no software burden to the CPU. The E^2 device is treated as any other memory element in the system. The reason for this lack of software requirement is the fact that all the burden is placed on the system hardware during the write time.



NOTES: (UNLESS OTHERWISE SPECIFIED)

1. RESET SIGNAL ORIGIN IS SYSTEM DEMONSTOR UNIT J1-22.
2. RESISTOR VALUES ARE IN OHMS, 1/4W, ± 5%.
3. +5V CONNECTED TO PIN 14 AND GROUND CONNECTED TO PIN 7 ON INTEGRATED CIRCUITS.
4. TEST POINTS
   1. GROUND
   2. CRTLEN
   3. READ
   4. READY
   5. ADDRESS 0
   6. DATA 0
   7. VCC
   8. OE CONTROL-READ
   9. RD
   10. VPP
5. ALL DIODES IN914.
6. ALL CAPACITORS IN μf.

AFN-01885A

Figure 5. E²-Demo Controller I

AFN-01913A

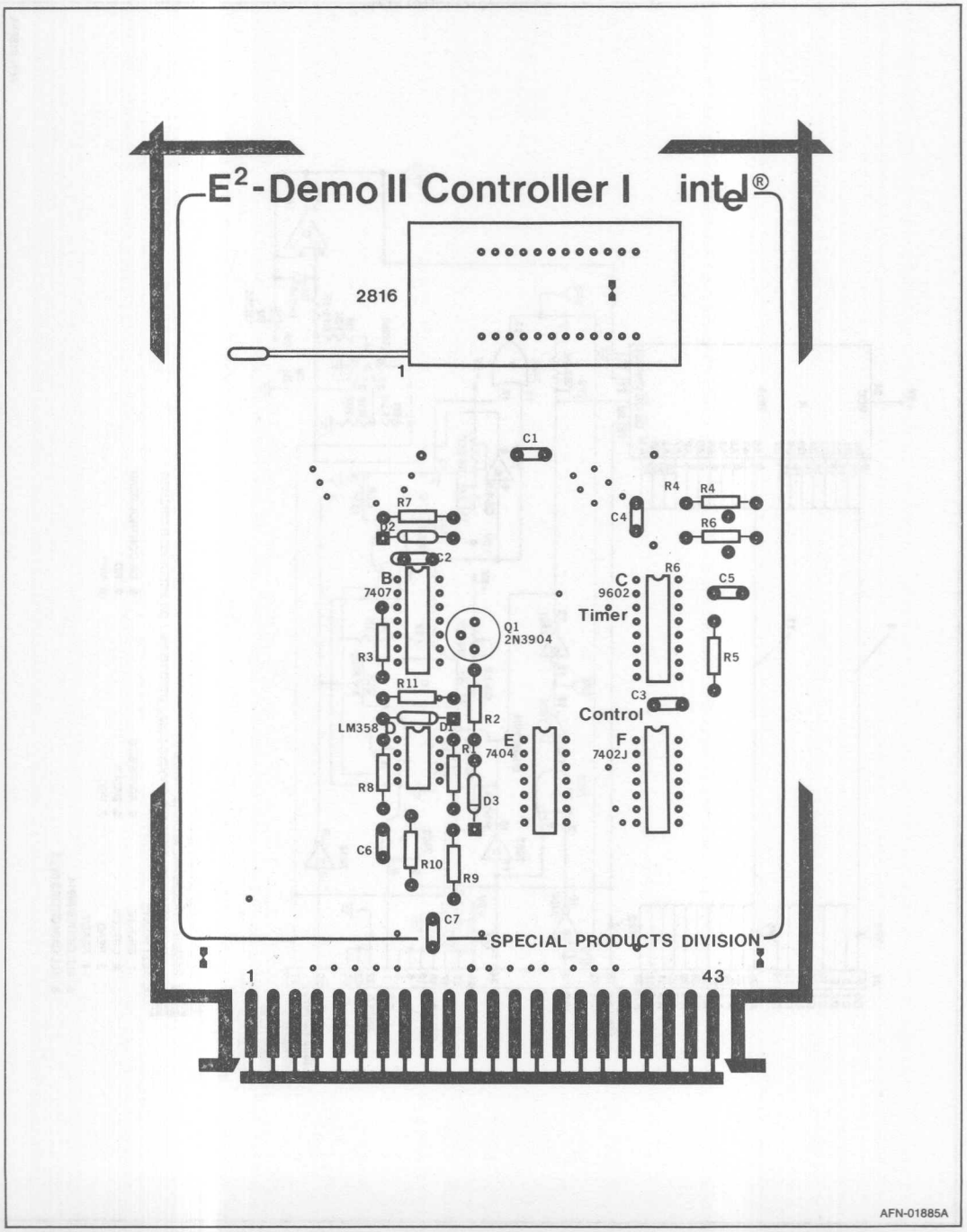


Figure 6a. E²-Demo II Controller I

AFN-01913A

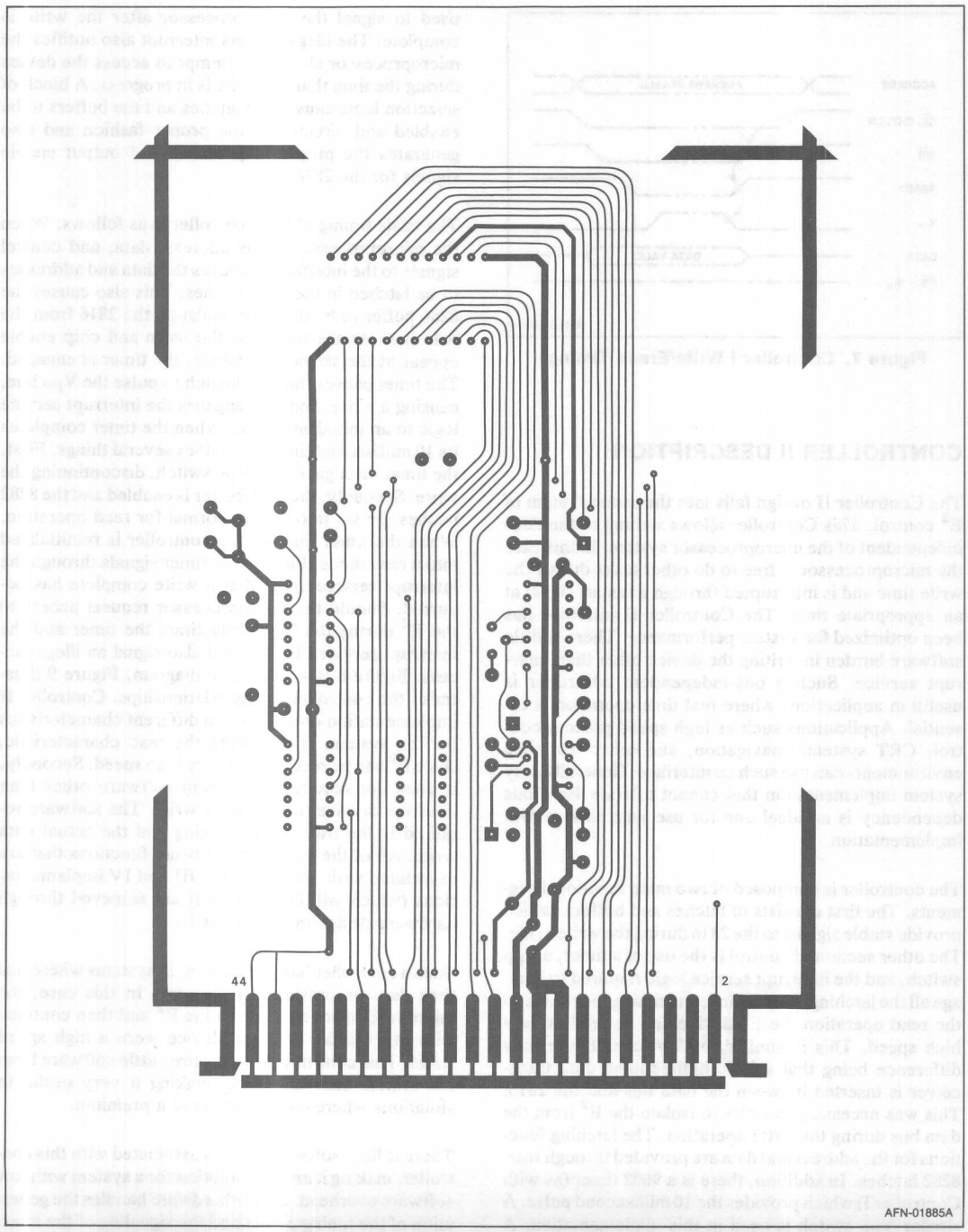


Figure 6b. E^2-Demo II Controller I (Continued)

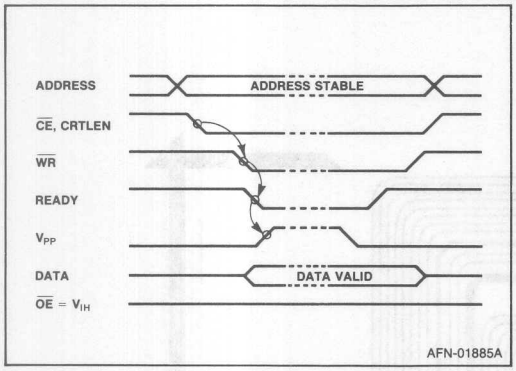


Figure 7. Controller I Write/Erase Timing

CONTROLLER II DESCRIPTION

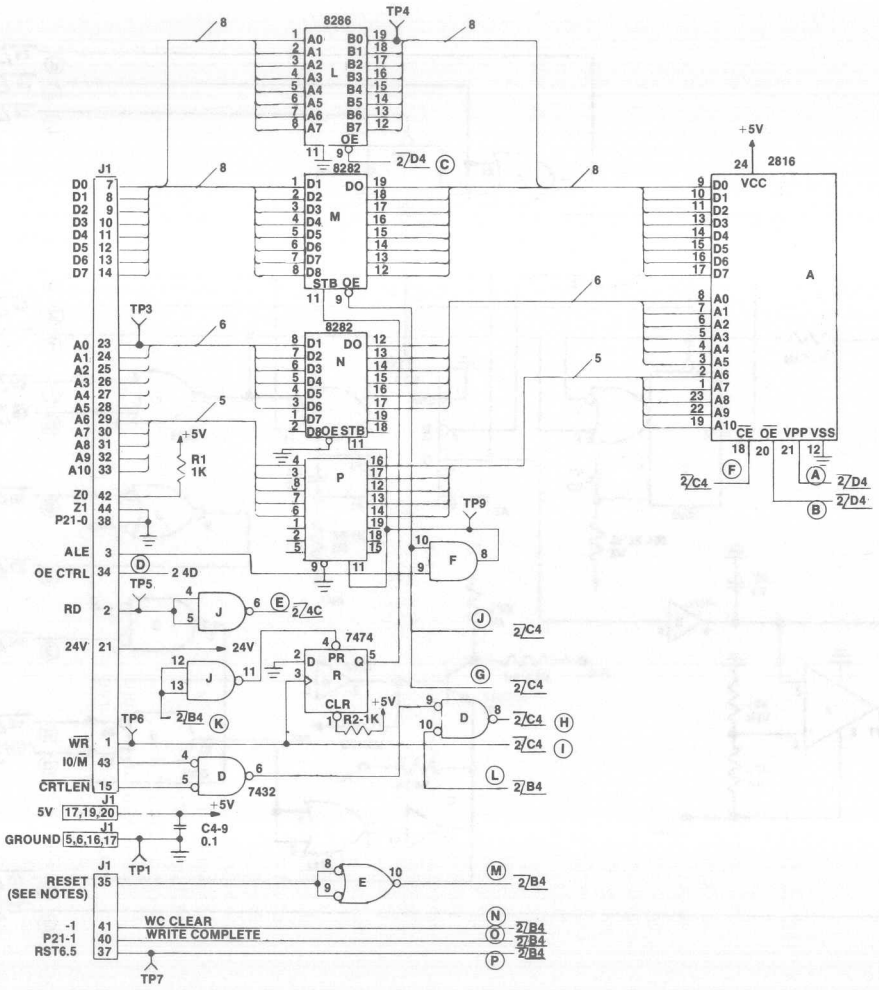
The Controller II design falls into the second realm of E^2 control. This Controller allows writing of the 2816 independent of the microprocessor system. In this case the microprocessor is free to do other tasks during the write time and is interrupted through a restart signal at an appropriate time. The Controller II interface has been optimized for system performance. There is little software burden in writing the device other than interrupt service. Such a bus-independent controller is useful in applications where real time operation is essential. Applications such as high speed process control, CRT systems, navigation, and other real time environments can use such an interface. Generally, any system implementation that cannot tolerate 10ms bus dependency is an ideal one for use with this control implementation.

The controller is composed of two main functional segments. The first consists of latches and buffers, which provide stable signals to the 2816 during the write cycle. The other section of control is the use of a timer, a V_{PP} switch, and the interrupt service logic required to manage all the latching, controlling, and timing functions. In the read operation the E^2 device can be read at very high speed. This is similar to Controller I, the only difference being that a 8286 bidirectional data transceiver is inserted between the data bus and the 2816. This was necessary in order to isolate the E^2 from the data bus during the write operation. The latching functions for the address and data are provided through Intel 8282 latches. In addition, there is a 9602 timer (as with Controller I) which provides the 10 millisecond pulse. A similar V_{PP} switch is used in this implementation. A block of interrupt service logic, which provides write complete interrupts and illegal-access interrupts, is used to signal the microprocessor after the write is complete. The illegal-access interrupt also notifies the microprocessor should it attempt to access the device during the time that the write is in progress. A block of selection logic causes the latches and the buffers to be enabled and directed in the proper fashion and also generates the proper chip enable and output enable signals for the 2816.

The basic timing of this controller is as follows: When the microprocessor sends address, data, and control signals to the interface, it causes the data and addresses to be latched in the 8282 latches. This also causes the 8286 buffer to be disabled, isolating the 2816 from the data bus. At the time that the write and chip enable appear at the select logic block, the timer is engaged. The timer causes the V_{PP} switch to pulse the V_{PP} line, causing a write, and also engages the interrupt service logic to an initialized state. When the timer completes its 10 millisecond time out, it does several things. First, the timer disengages the V_{PP} switch, discontinuing the write. Secondly, the 8286 buffer is enabled and the 8282 latches are set into a state normal for read operation. When the timer finishes the controller is reinitialized into a read mode. Finally, the timer signals through the interrupt service block that a write complete has occurred. Should the microprocessor request access to the E^2 during the long write time, the timer and the interrupt service block would also signal an illegal access. Figure 8 is a schematic diagram, Figure 9 illustrates the controller timing relationships. Controller II implementation optimizes two different characteristics for the system. It optimizes the read characteristic, since E^2 can be read from at very high speed. Secondly, it does not require any system software other than interrupt service to perform a write. The software required is the transparent erasing and the actual data write. All of the necessary software functions that are associated with the Controller III and IV implementations (which will be discussed) are achieved through hardware design in Controller II.

Such a controller has applications in systems where real time data processing is necessary. In this case, the microprocessor can write to the E^2 and then continue with other tasks as if the device were a high speed RAM. This controller also requires little software from the system software bank, making it very useful in situations where code space is at a premium.

There is little software burden associated with this controller, making it an ideal solution for a system with low software overhead. All the hardware handles the generation of the timing pulses and the signaling of the interrupt service at the proper time. Figure 10 shows the printed circuit layouts.



NOTES: (UNLESS OTHERWISE SPECIFIED)

1. RESET SIGNAL ORIGIN IS SYSTEM DEMONSTRATOR UNIT JI-22.
2. RESISTOR VALUES ARE IN OHMS, 1/4W, ±5%.
3. +5V CONNECTED TO PIN 14 AND GROUND CONNECTED TO PIN 7 ON INTEGRATED CIRCUIT.
4. TEST POINTS
   1. GROUND
   2. CE
   3. ADDRESS 0
   4. DATA
   5. READ
   6. WRITE
   7. RST65
   8. WRITE PULSE
   9. ALE-WRITE LOCKOUT
5. ALL DIODES IN914.
6. ALL CAPACITORS IN μf.

Figure 8a. E^2-Demo Controller II

AFN-01913A

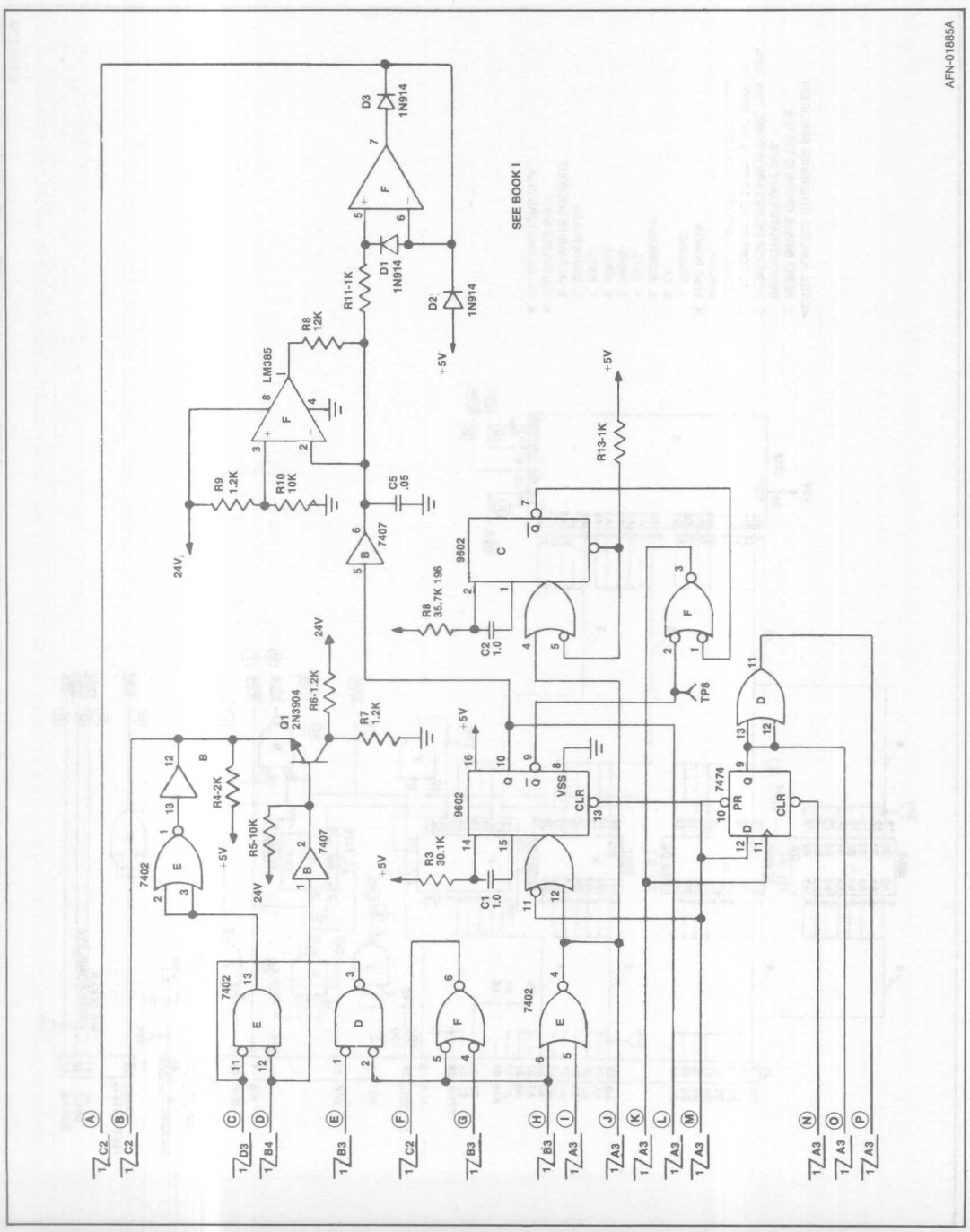


Figure 8b. E^2-Demo Controller II (Continued)

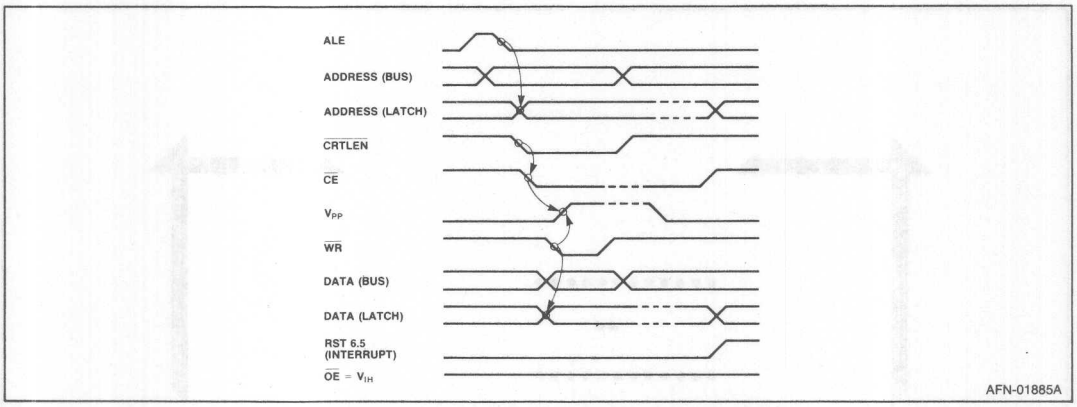


Figure 9. Controller II Write/Erase Timing

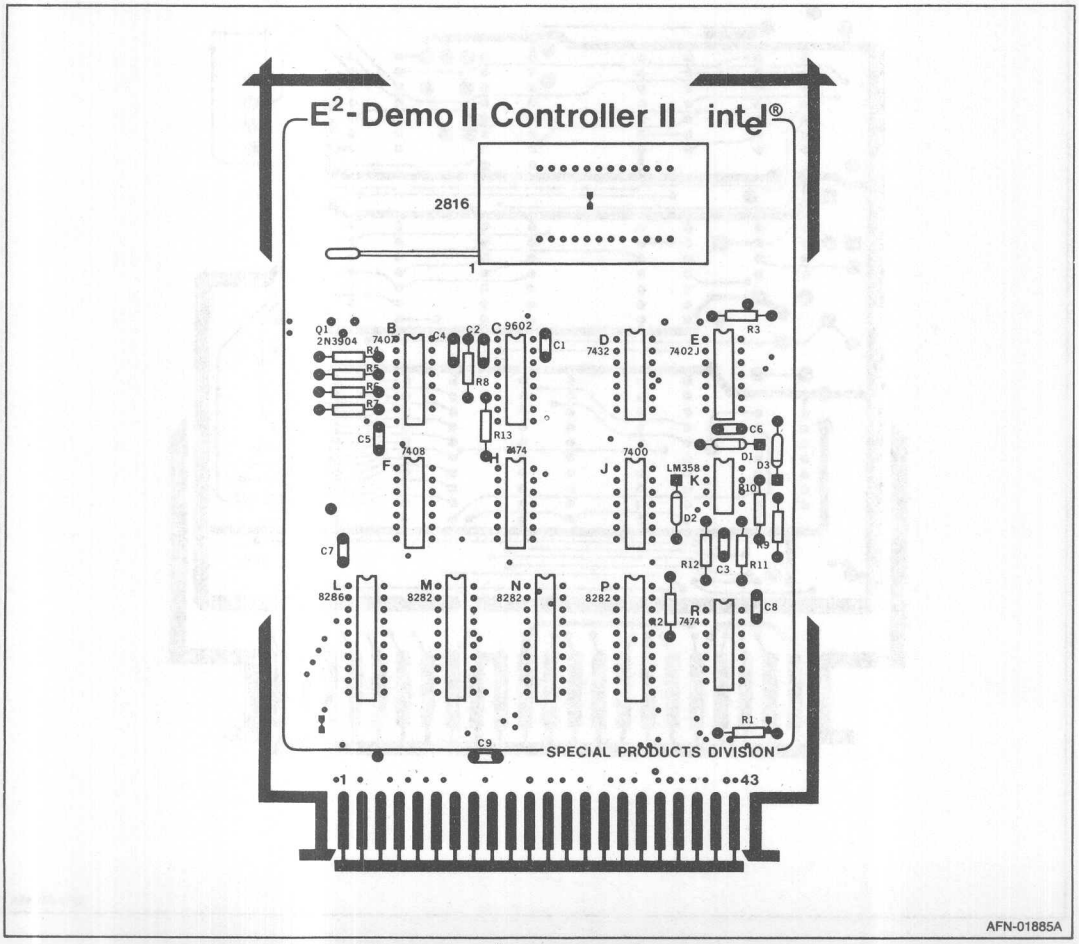


Figure 10a. E²-Demo II Controller II

AFN-01913A

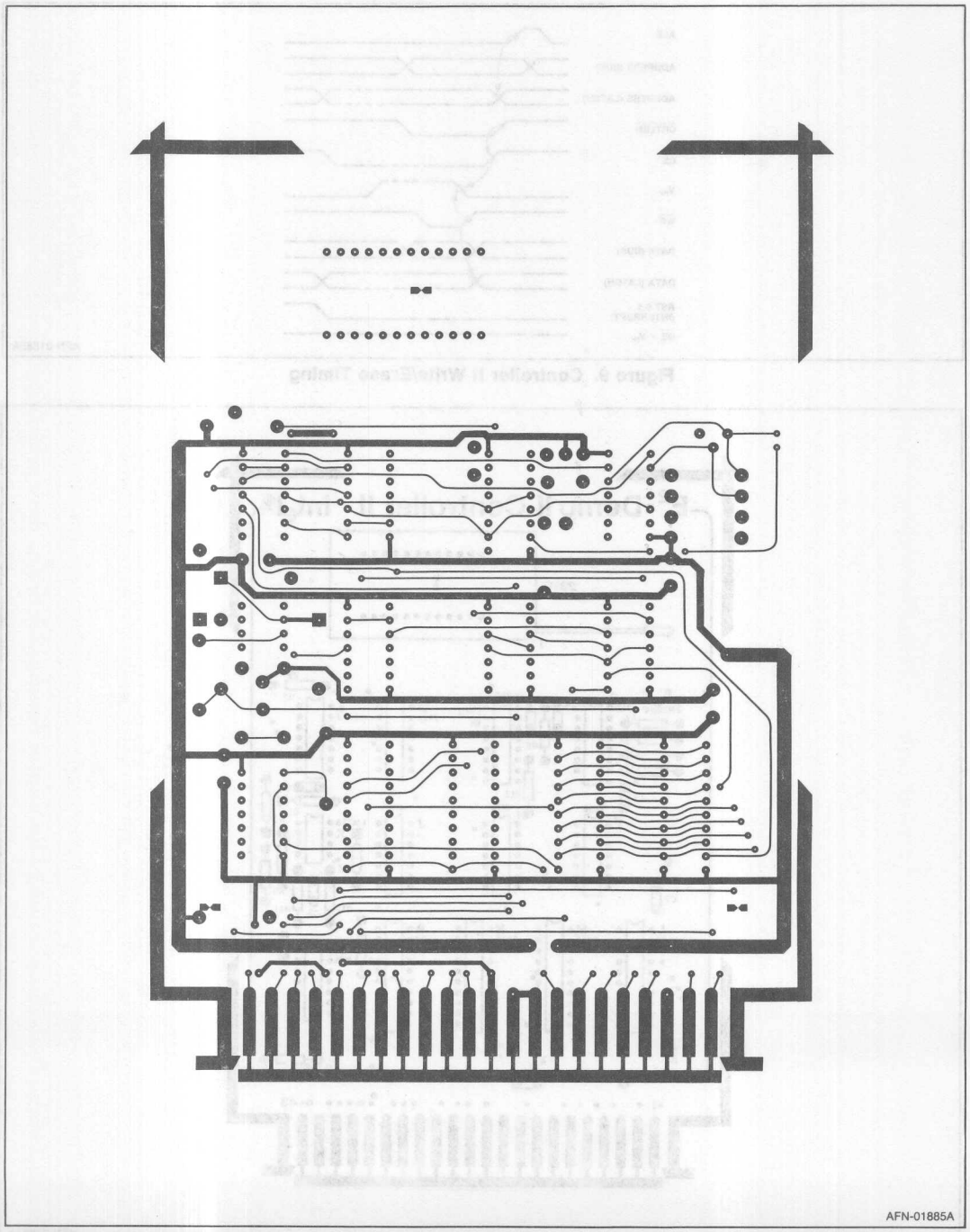


Figure 10b. E^2-Demo II Controller II (Continued)

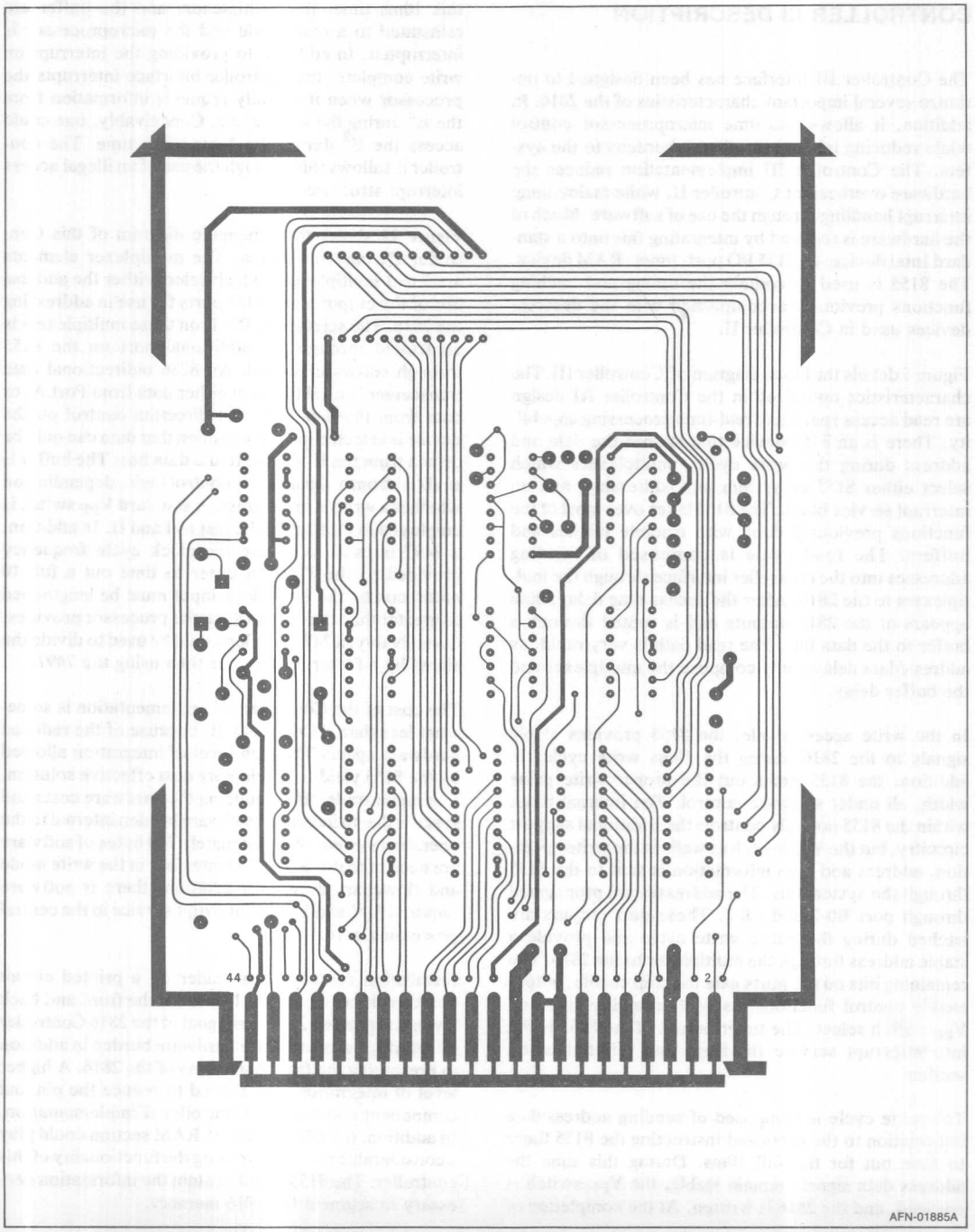


Figure 10c. E^2-Demo II Controller II (Continued)

CONTROLLER III DESCRIPTION

The Controller III interface has been designed to optimize several important characteristics of the 2816. In addition, it allows real-time microprocessor control while reducing inherent hardware burdens to the system. The Controller III implementation reduces the hardware overhead of Controller II, while maintaining interrupt handling through the use of software. Much of the hardware is reduced by integrating this onto a standard Intel device; an 8155 I/O port, timer, RAM device. The 8155 is used to contain the timing and latching functions previously accomplished with the descrete devices used in Controller II.

Figure 5 details the block diagram of Controller III. The characteristics optimized in the Controller III design are read access speed and real-time processing capability. There is an 8155 device that latches the data and address during the write cycle, multiplexers which select either 8155 or system bus addressing, and an interrupt service block. The 8155 takes over most of the functions previously done with discrete latches and buffers. The read cycle is composed of sending addresses into the controller interface through the multiplexers to the 2816. After the access time delay, data appears at the 2816 outputs and is routed through a buffer to the data bus. The read path is very rapid, as address/data delays only compose the multiplexer and the buffer delay.

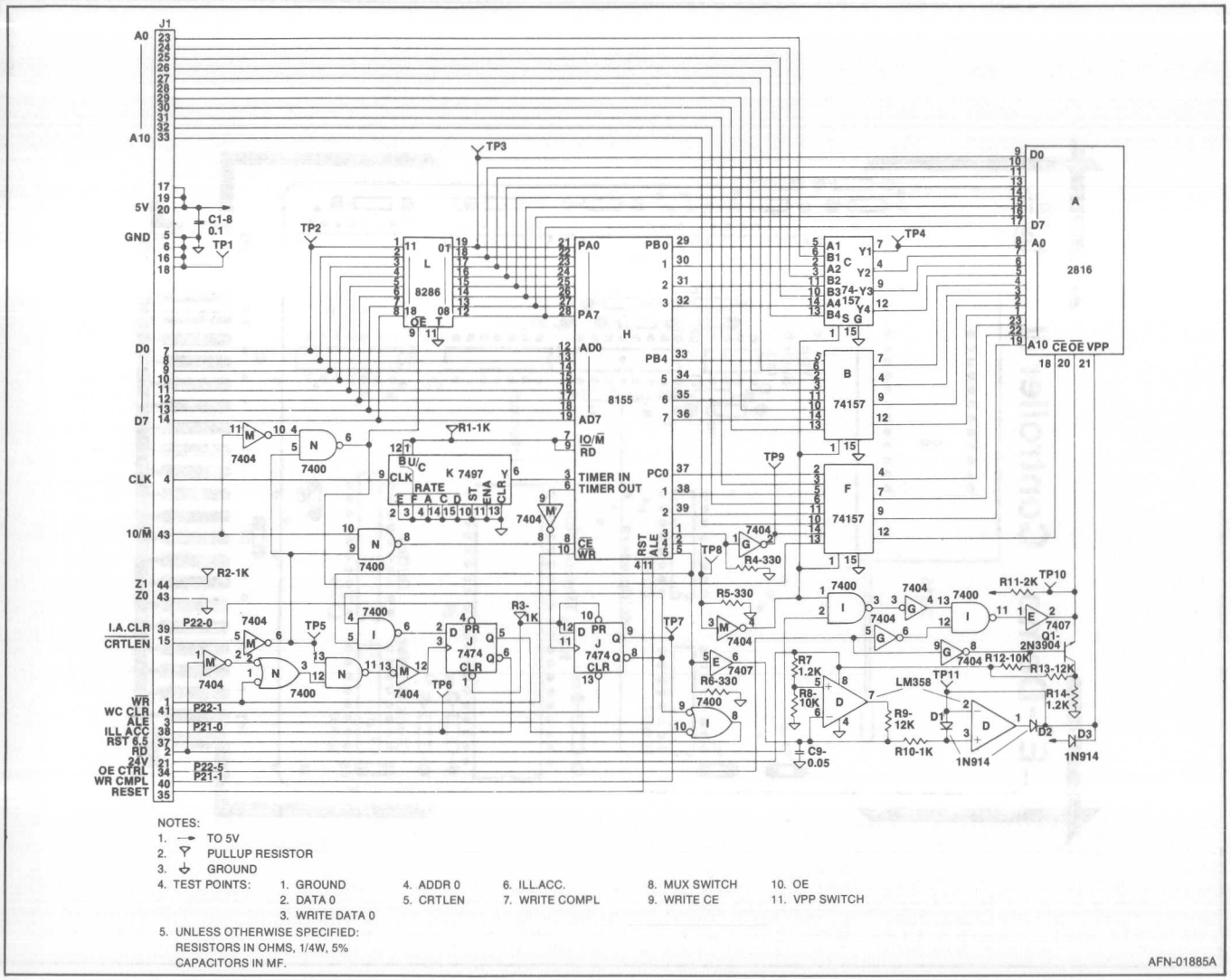
In the write access mode, the 8155 provides stable signals to the 2816 during the 10ms write cycle. In addition, the 8155 times out the proper write pulse width, all under software control. The internal timer within the 8155 not only controls the additional support circuitry, but the V_{PP} switch as well. In the write operation, address and data information is sent to the 8155 through the system bus. The addresses are propagated through port B0-7 and C0-2. These port outputs are latched during the entire write cycle and provide a stable address through the multiplexer to the 2816. The remaining bits on the ports gate the chip enable, output enable control functions, as well as multiplexer and V_{PP} switch select. The timer output of the 8155 is fed into interrupt service flip-flops and reinitialization section.

The write cycle is composed of sending address/data information to the ports and instructing the 8155 timer to time out for the full 10ms. During this time the address data signals remain stable, the V_{PP} switch is engaged, and the 2816 is written. At the completion of this 10ms time, the multiplexers and the buffer are reinstated to a read mode and the microprocessor is interrupted. In addition to providing the interrupt on write complete, the controller interface interrupts the processor when it illegally requests information from the E^2 during the write cycle. Conceivably, one could access the E^2 during the 10ms write time. The controller disallows this through the use of an illegal access interrupt structure.

Figure 11 shows the schematic diagram of this Controller III implementation. The multiplexer elements are 2 to 1 multiplexers, which select either the address bus or the output of the 8155 ports for use in addressing the 2816. The select line, Pin 1, on these multiplexers is controlled through the additional port on the 8155 through software control. An 8286 bidirectional data transceiver is used to select either data from Port A, or data from the data bus. The direction control on the device is selected in such a fashion that data can only be driven from the E^2 device to the data bus. The buffer is enabled from a signal in the control logic, depending on whether a write is in progress. A standard V_{PP} switch is employed in Controller III, just as I and II. In addition, a 7497 is used to reduce the clock cycle frequency provided to the 8155. In order to time out a full 10 milliseconds, the 8155 clock input must be lengthened to greater than the 320ns which the processor provides. Conceivably, a 7474 flip-flop could be used to divide the signal by a factor of 2, rather than using the 7497.

The cost of the Controller III implementation is somewhat less than a Controller II, because of the reduced hardware space. The high level of integration allowed by the 8155 yields a much more cost effective solution. The major trade-off in reducing the hardware costs and space is due to increased software burden internal to the operating system. Approximately 100 bytes of software are needed to drive the 8155 interface in the write mode and flowchart shown. In addition, there is software required for handling the interrupt service in the central processing core.

Installation of such a controller on a printed circuit board is shown in Figure 12, where the front and back layouts are shown. The main goal of the 2816 Controller III interface was to reduce hardware burden in addition to preserving the fast read access of the 2816. A higher level of integration was desired to reduce the pin and component count of the Controller II implementation. In addition, the use of the 8155 RAM section could play a considerable role in increasing the functionality of this controller. The 8155 could contain the information necessary to segment the 2816 memory.



NOTES:

1. TO 5V
2. PULLUP RESISTOR
3. GROUND
4. TEST POINTS:
   1. GROUND
   2. DATA 0
   3. WRITE DATA 0
   4. ADDR 0
   5. CRTLEN
   6. ILL.ACC.
   7. WRITE COMPL
   8. MUX SWITCH
   9. WRITE CE
   10. OE
   11. VPP SWITCH
5. UNLESS OTHERWISE SPECIFIED:
   RESISTORS IN OHMS, 1/4W, 5%
   CAPACITORS IN MF.

Figure 11. E²-Demo II Controller III

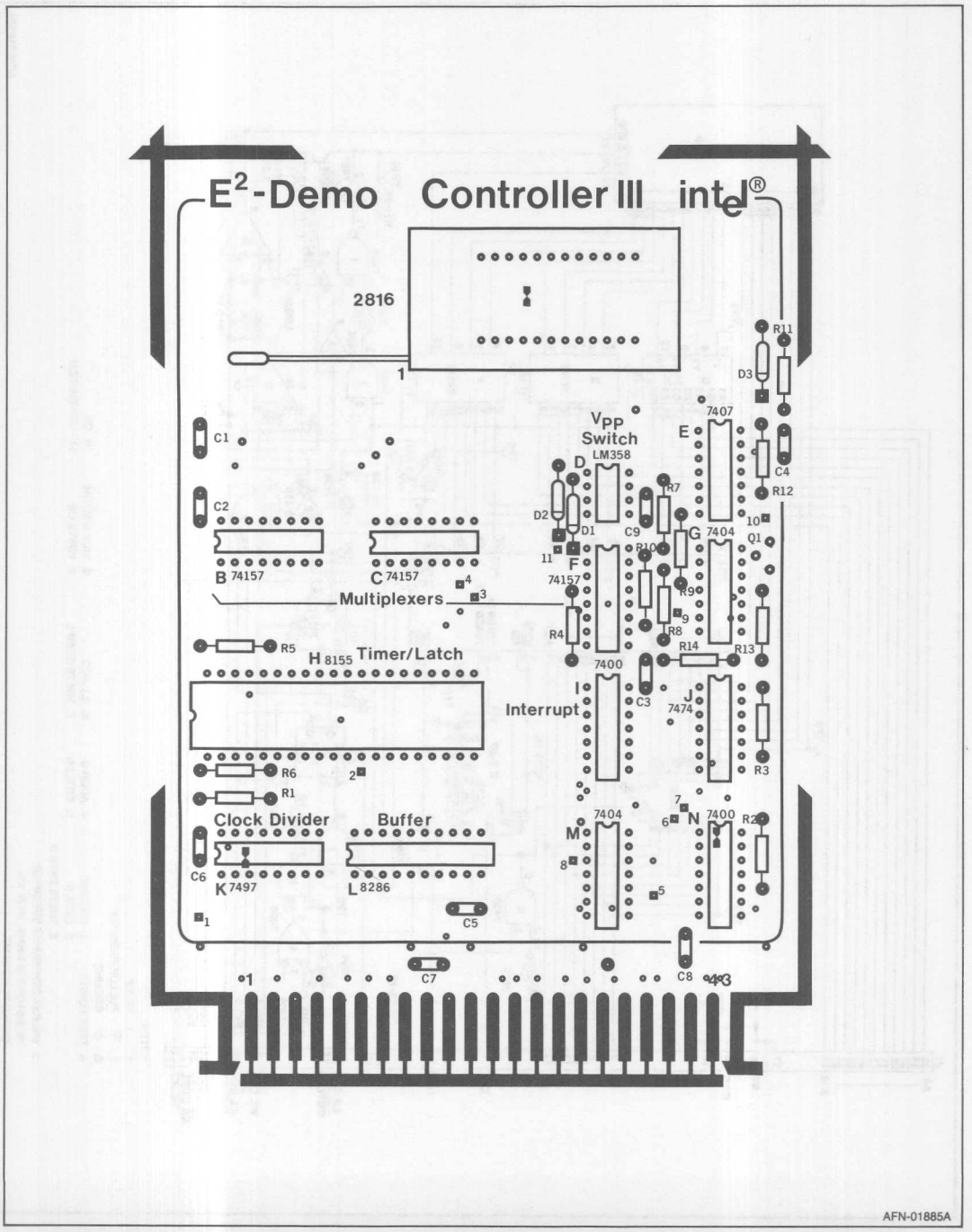
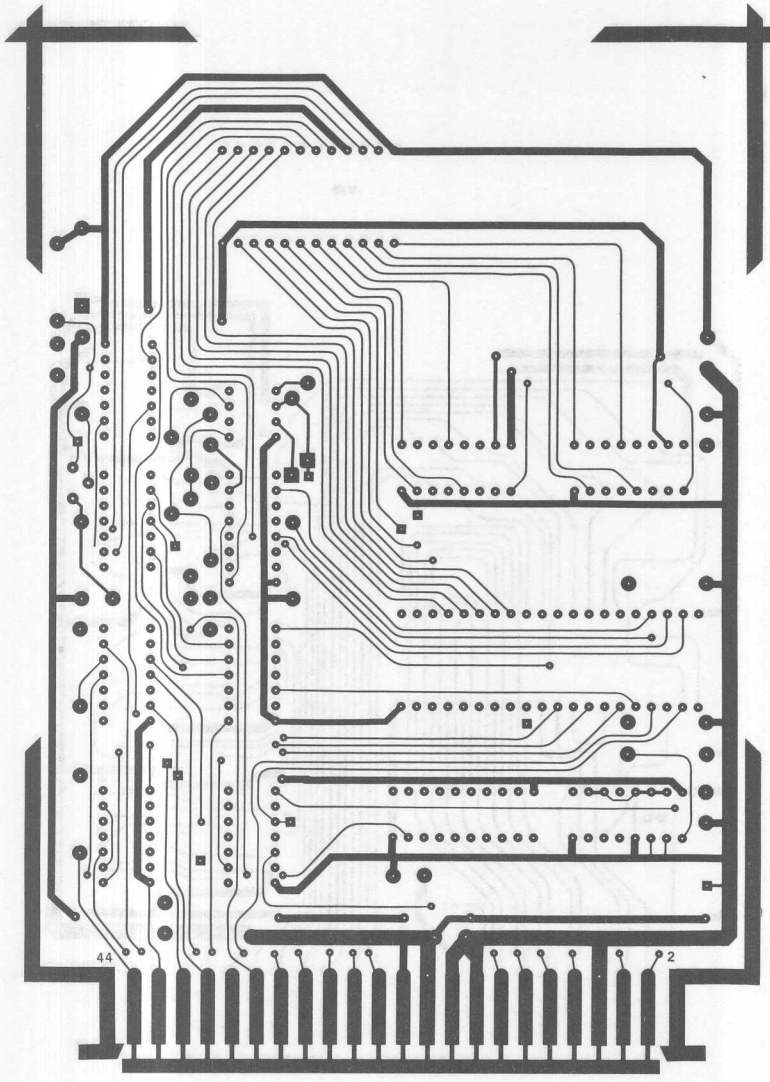


Figure 12a. E²-Demo Controller III

AFN-01913A



AFN-01885A

Figure 12b. E^2-Demo Controller III (Continued)

AFN-01913A

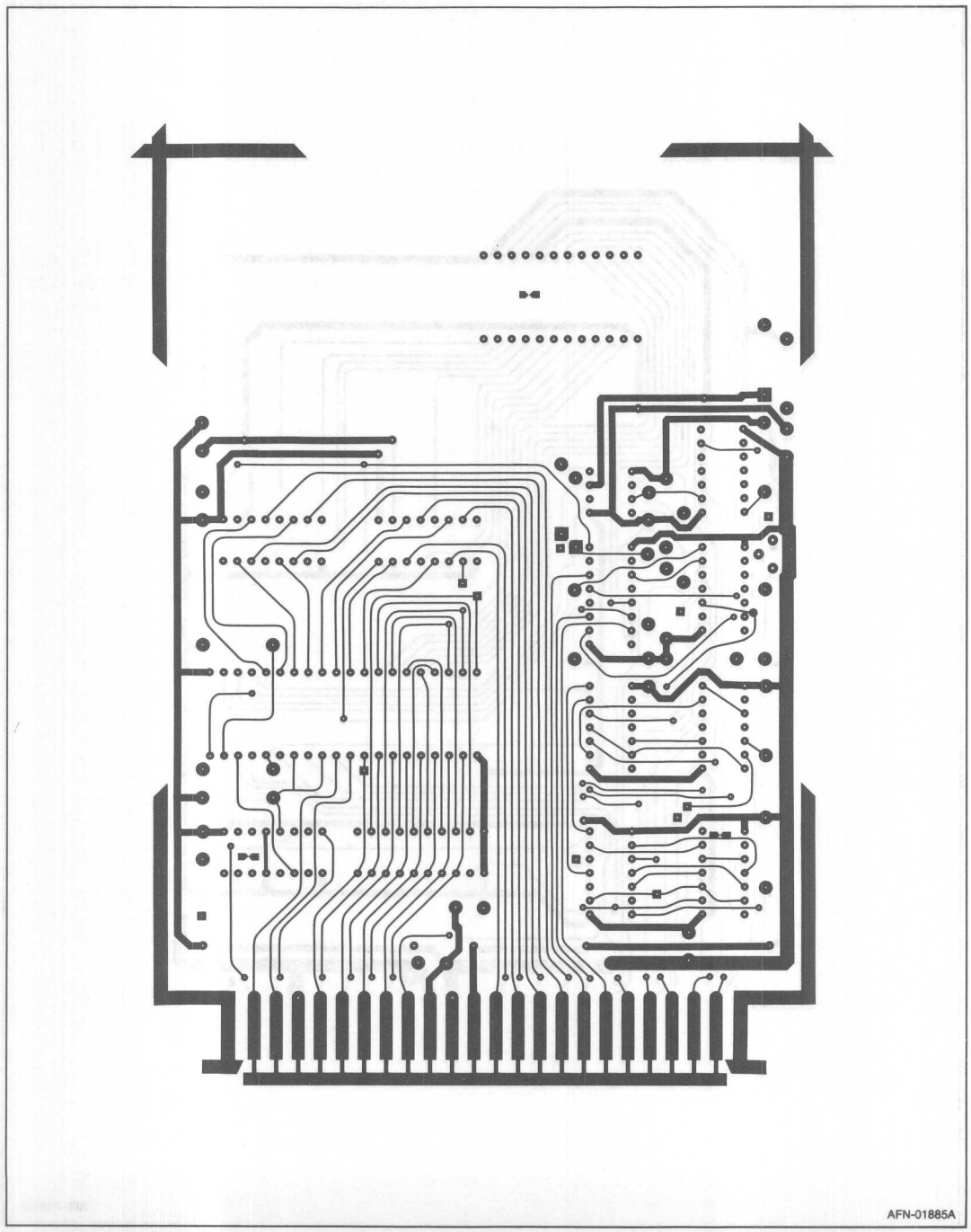


Figure 12c. E^2-Demo Controller III (Continued)

CONTROLLER IV DESCRIPTION

Data collection is the key with the Controller IV implementation. The interface described here was designed to accommodate those designs in a data-logging or data-store application mode. The constraints for such a design are small size, relatively low power, and a high level of integration. Those constraints that are not of concern in a data-store application are read access time, where write time may be critical. An attempt was made to reduce the hardware burden associated with a data logging application, while maintaining a relatively efficient write access interaction. The read access time is the parameter that has been compromised for this design. In this case we use an I/O port, timer chip, as before, to cause latching of the signals for the 2816. However, the 8155 is utilized for both read and write operations. To read from the 2816, address/data information is sent into the 8155.

Addresses are sent into the 2816 through Port B and C, data is read back out from Port A. Since the I/O ports on the 8155 can be configured in either input or output modes, we can use one set for addresses and the other set for data. Data is brought back from the 2816 through the 8155 and placed on the multiplexed address/data bus. In order to write to the 2816 address, a software routine is set up which maps into the 8155 port.

Writing is accomplished by sending the address information through the address data bus into the 2816 through Ports B and C. The data is sent into Port A and is held latched while the write is in progress. Port C3 controls the chip enable function. Output enable and V_{PP} drive are controlled by peripheral logic circuitry. To cause a write to the 2816, after the address/data information is loaded into the ports, the timer is commanded to time out. At the completion of the 10ms the processor is interrupted from the interrupt service block.

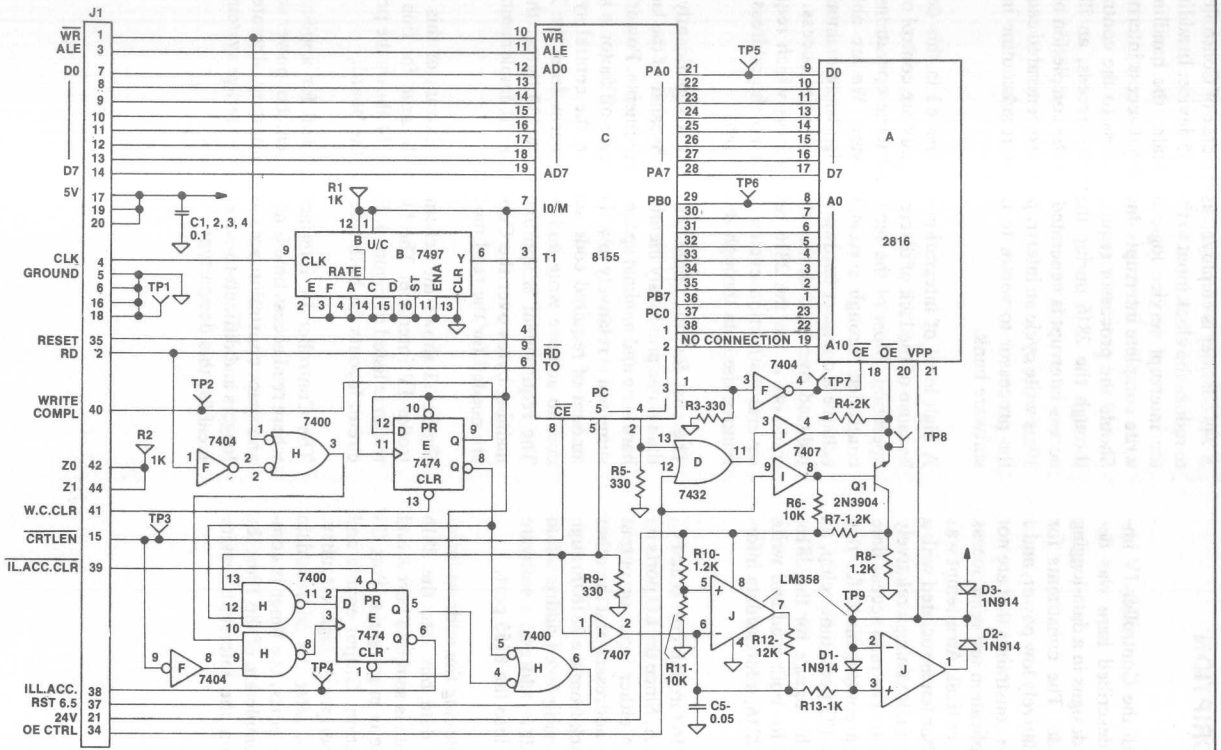
A 7497 divider is employed as the case of Controller III to reduce the clock input to the 8155 device. In addition, the interrupt service logic maintains the handling of write complete interrupts and illegal access interrupts. Should the processor request access to the controller through the 2816 during the write access, an illegal access interrupt is generated. At the completion of the 10ms write cycle an interrrupt is also generated causing the processor to vector to a restart subroutine in the software bank.

A high level of integration is achieved in this design because of the lack of discrete hardware control of the operation. Most of the read and write operations are controlled through system software. We are able to achieve a compact hardware design while maintaining reduced overhead during the 2816 write access. The trade-off is the the 2816 read access which requires several instruction cycles to set up the address and remove the data through the I/O port.

The cost for this implementation is significantly less than those previously mentioned because of the lack of hardware and minimal space requirements. Power consumption is relatively low. The trade-off factor is in the amount of required code space in the central system core to achieve write and read access from the 2816. The requirement is approximately 130 bytes, the remaining bytes over the Controller III implementation are needed for the read mode.

Figure 13 shows the schematic diagram of this Controller IV interface. The block diagram for this controller is listed in Figure 4. Figure 14 shows the printed circuit layouts for both sides of the board.

The Controller IV interface is ideal for applications where read access time is not critical, but power supply and space constraints are more important. Remote data loggers and difficult-to-access data storage systems are ideal for this design type.



NOTES:

1. → TO 5V
2. ▽ PULLUP RESISTOR
3. ⏚ GROUND
4. TEST POINTS:
   1. GROUND
   2. WRITE COMPLETE
   3. CRTLEN
   4. ILLEGAL ACCESS
   5. DATA 0
   6. ADDRESS 0
   7. CE
   8. OE
   9. VPP SWITCH
5. UNLESS OTHERWISE SPECIFIED:
   RESISTORS IN OHMS, 1/4W, 5%
   CAPACITORS IN MF.

AFN-01885A

Figure 13. Controller IV Schematic Diagram

AFN-01913A

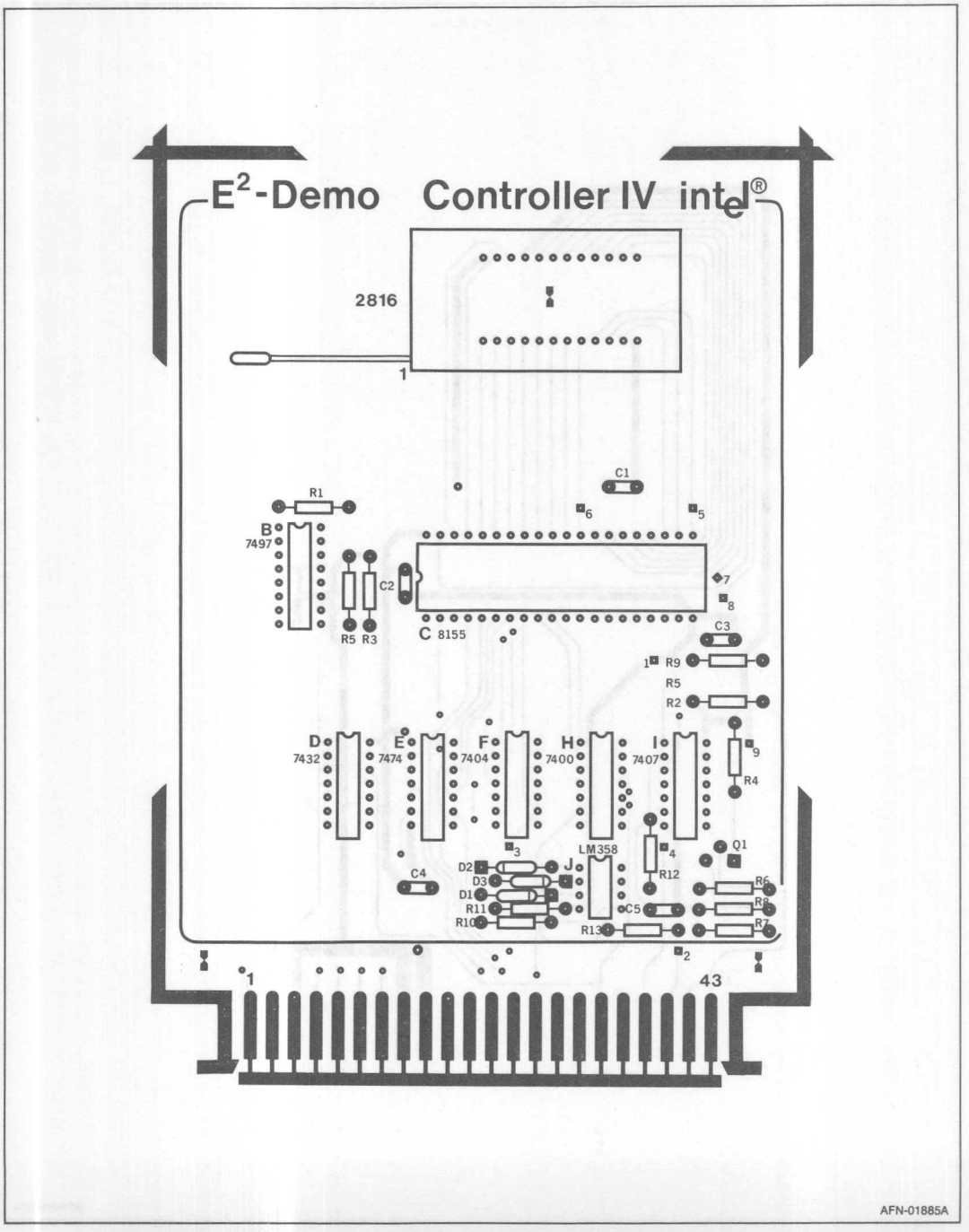


Figure 14a. E^2-Demo Controller IV

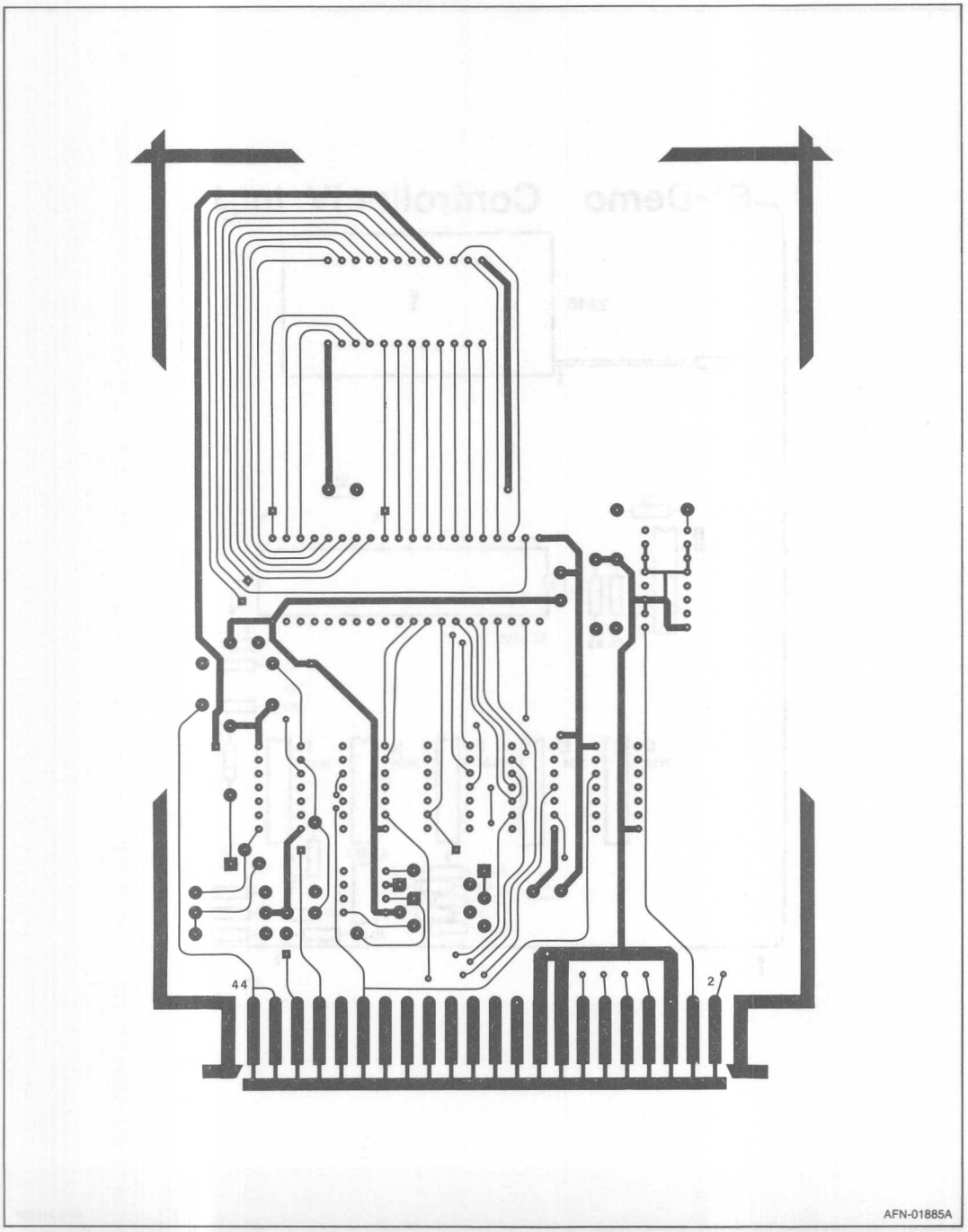


Figure 14b. E^2-Demo Controller IV (Continued)

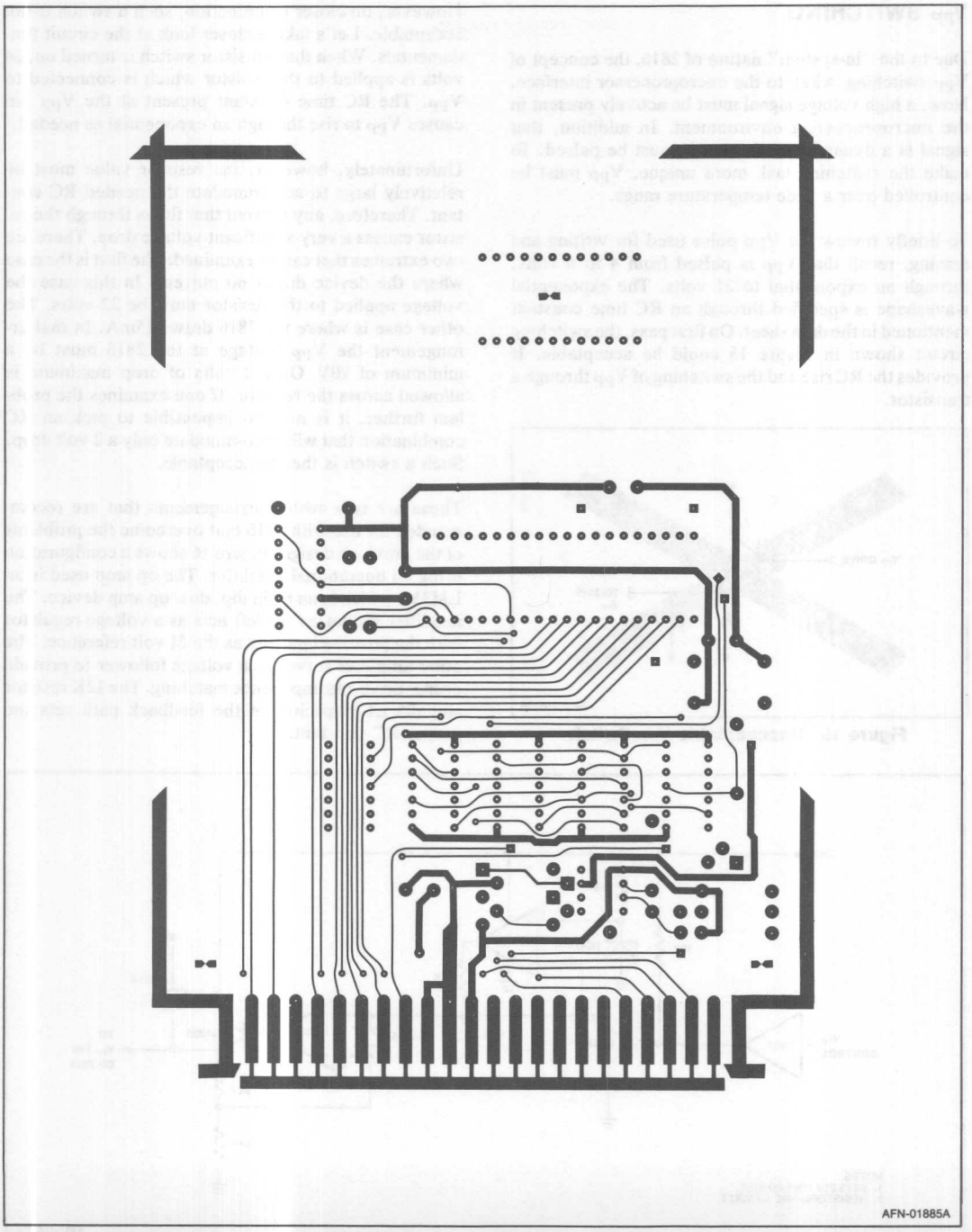


Figure 14c. E^2-Demo Controller IV (Continued)

V_{PP} SWITCHING

Due to the "in-system" nature of 2816, the concept of V_{PP} switching is key to the microprocessor interface. Now, a high voltage signal must be actively present in the microprocessing environment. In addition, that signal is a dynamic one in that it must be pulsed. To make the switching task more unique, V_{PP} must be controlled over a wide temperature range.

To briefly review the Vpp pulse used for writing and erasing, recall that V_{PP} is pulsed from 4 to 6 volts, through an exponential to 21 volts. The exponential waveshape is specified through an RC time constant mentioned in the data sheet. On first pass, the switching circuit shown in Figure 15 could be acceptable. It provides the RC rise and the switching of V_{PP} through a transistor.

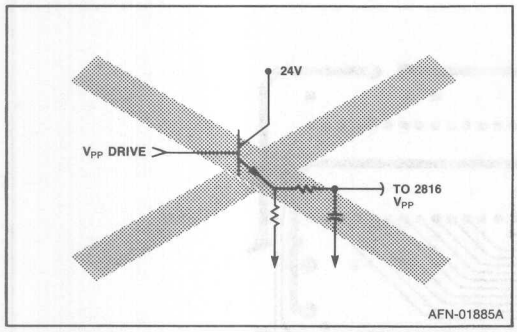


Figure 15. Unacceptable V_{PP} Switch

However, on closer examination, such a switch is not acceptable. Let's take a closer look at the circuit fundamentals. When the transistor switch is turned on, 24 volts is applied to the resistor which is connected to V_{PP}. The RC time constant present at the V_{PP} pin causes V_{PP} to rise through an exponential as needed.

Unfortunately, however, the resistor value must be relatively large to accommodate the needed RC contant. Therefore, any current that flows through the resistor causes a very significant voltage drop. There are two extremes that can be examined: The first is the case where the device draws no current. In this case the voltage applied to the resistor must be 22 volts. The other case is where the 2816 draws 15mA. In that arrangement the V_{PP} voltage at the 2816 must be a minimum of 20V. Only 2 volts of drop maximum is allowed across the resistor. If one examines the problem further, it is next to impossible to pick an RC combination that will accommodate only a 2 volt drop. Such a switch is then unacceptable.

These are two switch arrangements that are recommended for use with 2816 that overcome the problems of the previous design. Figure 16 shows a configuration using an operational amplifier. The op amp used is an LM358, which is an 8 pin dip, dual op amp device. The amplifier shown on the left acts as a voltage regulator with the positive input set as the 21 volt reference. The other amplifier serves as a voltage follower to provide proper drive and impedence matching. The 12K resistor and .05 μF capacitor in the feedback path sets the proper RC constant.

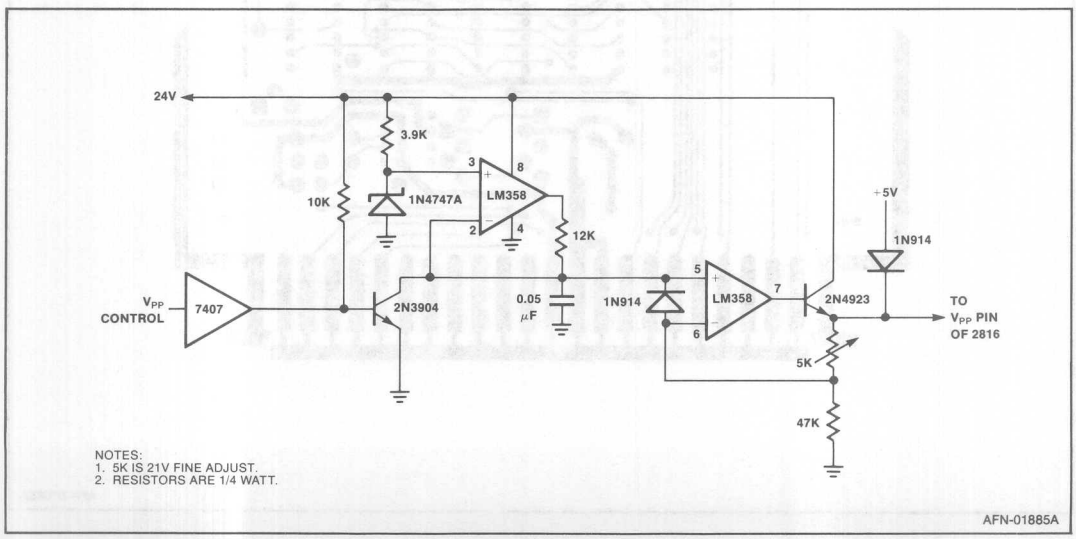


Figure 16. Operational Amplifier Switch

$\overline{OE}$ SWITCHING

The 2816, in addition to byte erase functionality, can implement chip erase. All 2048 bytes can be erased in only 10ms. To accomplish this, however, requires application of a high voltage, ultra-low current signal to the $\overline{OE}$ pin. When the output enable pin is set into the range of 9-15 volts, and the V_{PP} pin is pulsed to 21 volts, the entire chip is erased.

The current required at $\overline{OE}$ is a 10μA leakage, so little power is consumed. The switch shown in Figure 18 accomplishes switching $\overline{OE}$ to a higher voltage level when necessary. The chip erase control line can be derived from a port or other circuitry in the microprocessor system.

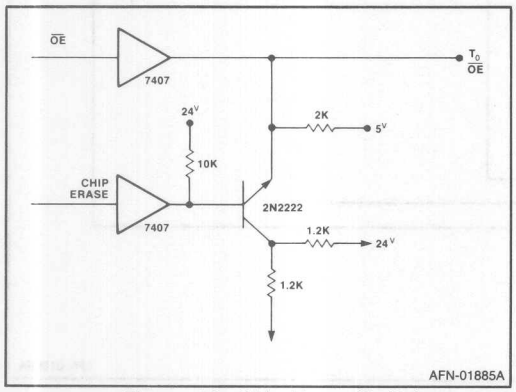


Figure 18. Chip Erase Switch

MULTIPLE 2816's

Because of the flexibility of E^2, the capability to easily connect multiple devices together is essential. RAM's can be simply tied together, E^2 needs a similar functionality. Figure 19 shows the mode select for the 2816's write/erase inhibit mode.

MODE \ PIN	$\overline{CE}$ (18)	$\overline{OE}$ (20)	V_{PP} (21)	OUTPUTS
READ	V_{IL}	V_{IL}	+4 to +6	D_{OUT}
STANDBY	V_{IH}	DON'T CARE	+4 to +6	HIGH Z
BYTE ERASE	V_{IL}	V_{IH}	+21	$D_{IN} = V_{IH}$
BYTE WRITE	V_{IL}	V_{IH}	+21	D_{IN}
CHIP ERASE	V_{IL}	+9 to +15V	+21	$D_{IN} = V_{IH}$
E/W INHIBIT	V_{IH}	DON'T CARE	DON'T CARE	HIGH Z

What this specification shows is that V_{PP} can be at high voltage (21V) when the 2816 is deselected. From a system perspective, V_{PP} can be bussed to multiple devices in the system. Any device that is to be written, can be, simply by TTL level control of $\overline{CE}$.

This allows simple and straightforward control of multiple 2816's in the system. Only one V_{PP} switch is needed for the entire memory array, allowing a highly compact and cost effective design. Figure 20 shows how simple such an implementation can be.

INTERFACE SOFTWARE REQUIREMENTS

As discussed, the various 2816 controllers employ various SSI and LSI devices. Each of the implementations require a varying degree of hardware and software. With Controller I, no software is necessary. Controller IV, on the other hand, needs approximately 130 bytes to handle the interface to the 8155 I/O port.

The following figures deal with the software drivers for the various controllers. These are several general subroutines that can be integrated in various ways, depending on the function and performance desired. Table 3 lists the various modules shown in the figures.

Table 3.

Overall Write Subroutine	Figure 21
Controller I Software Driver	Figure 22
Controller II Software Driver	Figure 23
Controller III Software Driver	Figure 24
Controller IV Software Driver	Figure 25
Controller II Chip Erase Routines	Figure 26
Controller III, IV Chip Erase Routines	Figure 27
Controller I/O Poll Routines	Figure 28
Controller Interrupt Driver	Figure 29

Figure 21 shows the generalized write subroutine for all controllers. As indicated, data is passed through the 8085 A-register, and addresses passed through the HL-register pair. The routine first executes an erase, and then a write operation. The software driver that writes to the device is called WECYCL.

There is a unique WECYCL routine for each control interface. The driver for Controller I is a simple parameter pass routine, and a move to memory. This software is listed in Figure 22. The Controller II subroutine uses parameter pass, and interrupt initialization and service. The Controller II driver is listed in Figure 23. The interrupt service routine is given in Figure 29. In order to write to Controller III and IV interfaces, the 8155 I/O device must be initialized. A generalized flow chart for this operation is shown in Figure 24A. The software

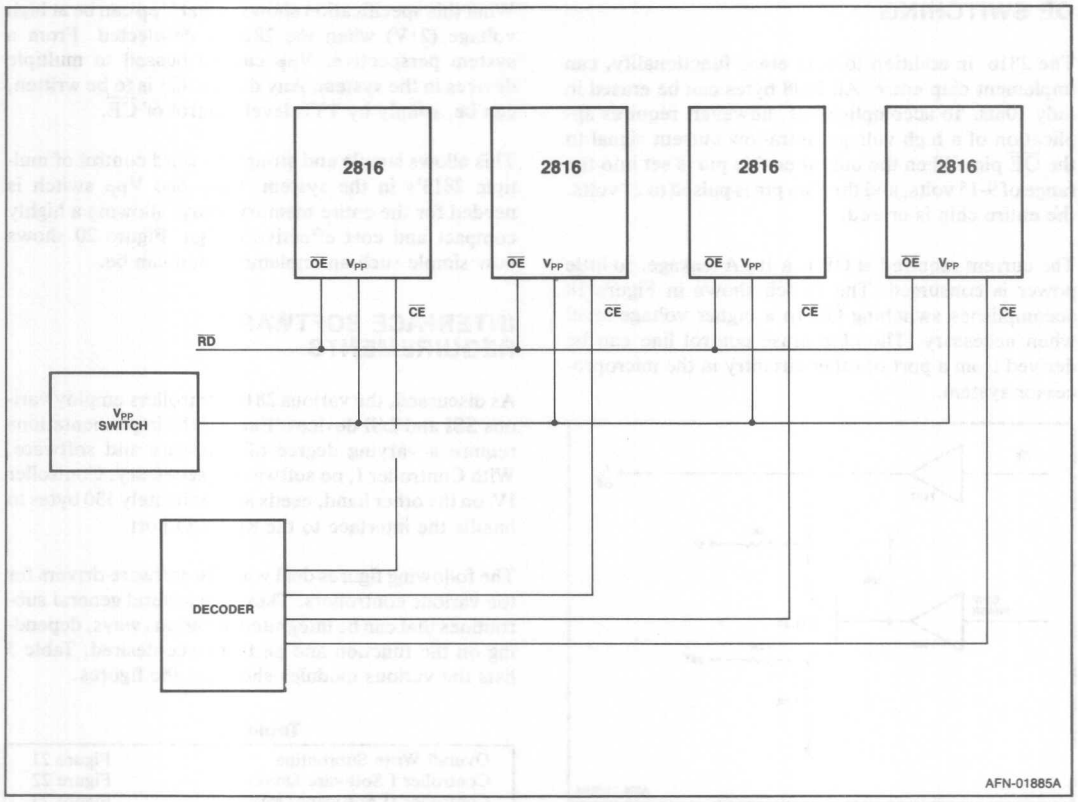


Figure 20. Multiple 2816's In System

listings are detailed in Figures 24 and 25. Both of these routines use the same interrupt service as Controller II. The remaining routines, for chip erase and I/O polling control, are shown in Figures 27, 28.

All of the interfaces, with the exception of the Controller IV, allow transparent reads of the 2816. Controller IV isolates the E^2 from the system bus through the 8155. A flowchart for Controller IV read operations is detailed in Figure 30.

CONCLUSION

Based on the previous discussion, it is apparent that the interface to the 2816 is highly application dependent. Several interfaces have been presented, each of those optimized for a different system concern. Each of the controller implementations requires a different amount of hardware and software overhead, and provides a different throughput capability to the host processor. Each of these controllers is also appropriate for one or more design types. Controller I for program store areas, Controller IV for strict data store applications.

Controllers II and III are higher performance, and yet require a larger amount of hardware and software to service interrupts and generate 8155 timing controls. Further application notes will discuss some of the enhanced controllers, such as the bipolar state machine controller. All of these controllers are also available for test in the E^2 Demonstrator, which is a highly sophisticated tool for use with the 2816. The demonstrator is available by contacting a local Intel sales office and can be used for evaluation and test purposes of the E^2 device.

Above all, the interface to the CPU has been realized in a consistent and appropriate microprocessing architecture, something that has never been possible because of prior device attributes and technology constraints. The 2816 then adds an appropriate and applicable use of non-volatile and flexible memory to the current offerings of memory devices. It will prove a useful and powerful memory supplement and yield application and system benefits never before possible through consistent, convenient, and simple microprocessor interface.

AFN-01913A

```
ASM80 :F1:WRITE.SRC

ISIS-II 8080/8085 MACRO ASSEMBLER, V3.0        MODULE   PAGE    1

  LOC  OBJ         LINE       SOURCE STATEMENT

                      1 $DEBUG
                      2
                      3
                      4
                      5         ;       ********************************************
                      6
                      7
                      8         ;               2816 CONTROLLER WRITE SUBROUTINE
                      9
                     10         ;
                     11         ;
                     12
                     13         ;       ********************************************
                     14
                     15
                     16         EXTRN   PECYCL
                     17
                     18         PUBLIC  WRITE
                     19
                     20         CSEG
                     21
                     22
                     23         ;       WRITE SUBROUTINE
                     24
                     25         ;       WRITES A BYTE TO THE 2816
                     26
                     27         ;       DATA PASSED:    A = DATA TO WRITE
                     28         ;                       HL = ADDRESS TO WRITE
                     29         ;       REGS DESTROYED: NONE
                     30         ;       CALLS:          PECYCL - PROGRAM/ERASE CYCLE SUBROUTINE
                     31
                     32 WRITE:
  0000 F5            33         PUSH    PSW             ; SAVE DATA WE'RE ABOUT TO WRITE
  0001 3EFF          34         MVI     A,0FFH          ; EXECUTE A BYTE ERASE FUNCTION
  0003 CD0000    E   35         CALL    PECYCL          ; BY WRITING 0FFH TO THE 2816
  0006 F1            36         POP     PSW             ; RESTORE DATA BYTE
  0007 CD0000    E   37         CALL    PECYCL          ; NOW WRITE THE DATA
  000A C9            38         RET                     ; AND RETURN
                     39
                     40         END

PUBLIC SYMBOLS
WRITE  C 0000

EXTERNAL SYMBOLS
PECYCL E 0000

USER S   JLS
PECYCL E 0000    WRITE  C 0000

ASSEMBLY COMPLETE,   NO ERRORS                                          AFN-01885A
```

Figure 21. Overall Write Subroutine

AFN-01913A

```
ASM80 :F1:CONT1.SRC MOD85

ISIS-II 8080/8085 MACRO ASSEMBLER, V3.0        MODULE   PAGE    1


  LOC  OBJ        LINE       SOURCE STATEMENT
                     1 $DEBUG
                     2
                     3
                     4        CSEG
                     5
                     6        PUBLIC  READ,WECYCL
                     7
                     8
                     9
                    10        ;       CONTROLLER I READ SUBROUTINE
                    11
                    12        ;       DATA PASSED:    HL = ADDRESS OF 2816 LOCATION TO READ
                    13        ;       DATA RETURNED:  A  = DATA READ
                    14        ;       REGS DESTROYED: NONE
                    15
                    16 READ:
  0000 7E           17        MOV     A,M             ; JUST READ FROM MEMORY
  0001 C9           18        RET
                    19
                    20
                    21
                    22
                    23        ;       CONTROLLER I WRITE/ERASE CYCLE SUBROUTINE
                    24
                    25        ;       DATA PASSED:    HL = ADDRESS OF 2816 LOCATION TO WRITE
                    26        ;                       A  = DATA TO WRITE
                    27        ;                            OR 0FFH  (ERASE)
                    28        ;       DATA RETURNED:  NONE
                    29        ;       REGS DESTROYED: NONE
                    30
                    31 WECYCL:
  0002 77           32        MOV     M,A             ; JUST WRITE TO MEMORY
  0003 C9           33        RET
                    34
                    35
                    36        END

PUBLIC SYMBOLS
READ   C 0000    WECYCL C 0002
EXTERNAL SYMBOLS

USER SYMBOLS
READ   C 0000    WECYCL C 0002
ASSEMBLY COMPLETE,   NO ERRORS
```

AFN-01885A

Figure 22. Controller I Software Driver

AFN-01913A

```
ASM80 :F1:CONT2.SRC MOD85
ISIS-II 8080/8085 MACRO ASSEMBLER, V3.0        MODULE   PAGE    1

  LOC  OBJ       LINE     SOURCE STATEMENT
                    1 $DEBUG
                    2
                    3
                    4         PUBLIC  WECYCL,READ,ENDWE,CCLEAR
                    5
                    6         EXTRN   WEDELY
                    7
                    8         CSEG
                    9
                   10
                   11         ;       CONTROLLER II READ SUBROUTINE
                   12
                   13         ;       DATA PASSED:    HL = ADDRESS OF 2816 LOCATION TO READ
                   14         ;       DATA RETURNED:  A  = DATA READ
                   15         ;       REGS DESTROYED: NONE
                   16
                   17 READ:
  0000 7E          18         MOV     A,M             ; JUST READ FROM MEMORY
  0001 C9          19         RET
                   20
                   21
                   22
                   23
                   24         ;       CONTROLLER II WRITE/ERASE CYCLE SUBROUTINE
                   25
                   26         ;       DATA PASSED:    HL = ADDRESS OF 2816 LOCATION TO WRITE
                   27         ;                       A  = DATA TO WRITE
                   28         ;                                  OR 0FFH  (ERASE)
                   29         ;       DATA RETURNED:  NONE
                   30         ;       REGS DESTROYED: NONE
                   31         ;       CALLS:          WEDELY (I/O POLL ROUTINE OR INTERRUPT DRIVER)
                   32
                   33         ;       I/O PORTS USED:
                   34         ;               PORT 22H (OUTPUT) - CONTAINS BITS USED FOR RESETTING
                   35         ;                                   CONTROLLER INTERRUPT FLIP/FLOPS
                   36         ;                       BIT 0 = WRITE COMPL RESET (ACTIVE LOW)
                   37         ;                       BIT 1 = ILL ACCESS  RESET (ACTIVE LOW)
                   38
                   39         ;       COMMENTS:       ENDWE (END OF WRITE/ERASE CYCLE) ROUTINE
                   40         ;                       IS CALLED BY INTERRUPT DRIVER OR I/O POLL
                   41         ;                       ROUTINE (WEDELY) TO SHUT DOWN CONTROLLER.
                   42         ;                       THIS SUBROUTINE IS PART OF THE DRIVER
                   43         ;                       PACKAGE ROUTINES INITIATED BY A CALL TO
                   44         ;                       WECYCL.
                   45
                   46
                   47         ;       I/O SYMBOLS
                   48
  0022             49 CLRPRT  EQU     22H             ; I/O PORT USED TO CLEAR INTERRUPT F/F'S
  0000             50 CLRACT  EQU     0               ; BIT PATTERN TO ACTIVATE CLEAR FUNCTION
```

AFN-01885A

Figure 23. Controller II Software Driver

AFN-01913A

```
ISIS-II 8080/8085 MACRO ASSEMBLER, V3.0        MODULE   PAGE    2
 LOC  OBJ         LINE       SOURCE STATEMENT

 0003              51 CLRINA  EQU     3H             ; BIT PATTERN TO DE-ACTIVATE CLEAR FUNCTION
                   52
                   53 WECYCL:
 0002 F5           54         PUSH    PSW            ; SAVE DATA TO WRITE
 0003 3E00         55         MVI     A,CLRACT       ; CLEAR WRITE COMPLETE AND ILLEGAL ACCESS F/F'S
 0005 D322         56         OUT     CLRPRT
 0007 3E03         57         MVI     A,CLRINA       ; DE-ACTIVATE CLEAR FUNCTION
 0009 D322         58         OUT     CLRPRT
 000B F1           59         POP     PSW            ; RESETORE DATA TO WRITE
 000C 77           60         MOV     M,A            ; JUST WRITE TO MEMORY
 000D CD0000   E   61         CALL    WEDELY         ; GO TO I/O POLL ROUTINE OR INTERRUPT DRIVER
 0010 C9           62         RET
                   63
                   64
                   65
                   66
                   67         ;       CONTROLLER II CHIP CLEAR SUBROUTINE
                   68
                   69         ;       DATA PASSED:    NONE
                   70         ;       DATA RETURNED:  NONE
                   71         ;       REGS DESTROYED: NONE
                   72         ;       CALLS           PEDELY ( I/O POLL ROUTINE OR INTERRUPT DRIVER)
                   73
                   74         ;       I/O PORTS USED:
                   75         ;                       PORT 22H (OUTPUT)
                   76         ;                               BIT 0 = WRITE COMPLETE CLEAR (ACTIVE LOW)
                   77         ;                               BIT 1 = ILLEGAL ACCESS CLEAR (ACTIVE LOW)
                   78         ;                               BIT 5 = CHIP CLR (+12V TO OE' LINE) (ACTIVE HI)
                   79
                   80         ;       COMMENTS:       ENDWE (END OF WRITE/ERASE CYCLE) ROUTINE
                   81         ;                       IS CALLED BY INTERRUPT DRIVER OR I/O POLL
                   82         ;                       ROUTINE (WEDELY) TO SHUT DOWN CONTROLLER.
                   83         ;                       THIS SUBROUTINE IS PART OF THE DRIVER
                   84         ;                       PACKAGE ROUTINES INITIATED BY A CALL TO
                   85         ;                       CCLEAR.
                   86
                   87
                   88         ;       I/O SYMBOLS
                   89
 0023              90 CLRCCL  EQU     23H            ; DATA TO DEACTIVATE CLEAR WC & IA BUT ACTIVATE
                   91                                ; OE' = +12V FUNCTION FOR CHIP CLEAR
                   92 CCLEAR:
 0011 F5           93         PUSH    PSW            ; SAVE REGISTERS
 0012 E5           94         PUSH    H
 0013 3E00         95         MVI     A,CLRACT       ; GET BITS TO RESET WRITE COMPL AND ILL ACC
 0015 D322         96         OUT     CLRPRT
 0017 3E23         97         MVI     A,CLRCCL       ; GET BITS TO DEACTIVATE CLEAR FUNCTION AND
                   98                                ; TURN ON OE' = +12V FUNCTION FOR CHIP CLEAR
 0019 D322         99         OUT     CLRPRT         ; OUTPUT TO I/O PORT
 001B 3EFF        100         MVI     A,0FFH         ; WRITE 0FFH TO THE 2816
 001D 3200A0      101         STA     0A000H                                          AFN-01885A
```

Figure 23. Controller II Software Driver (Continued)

AFN-01913A

```
ISIS-II 8080/8085 MACRO ASSEMBLER, V3.0        MODULE   PAGE   3
 LOC  OBJ         LINE       SOURCE STATEMENT

 0020 CD0000   E   102         CALL    WEDELY          ; GO TO I/O POLL LOOP OR INTERRUPT DRIVER
 0023 3E03         103         MVI     A,CLRINA        ; DEACTIVATE CHIP CLEAR FUNCTION
 0025 D322         104         OUT     CLRPRT
 0027 E1           105         POP     H               ; RESTORE REGISTERS
 0028 F1           106         POP     PSW
 0029 C9           107         RET
                   108
                   109
                   110
                   111         ;       CONTROLLER II END-OF-WRITE/ERASE-CYCLE ROUTINE
                   112
                   113         ;       JUMPED TO BY I/O POLL OR INTERRUPT DRIVER AFTER WRITE COMPLETE
                   114         ;       TO SHUT DOWN CONTROLLER.
                   115
                   116 ENDWE:
 002A C9           117         RET             ; JUST RETURN NORMALLY - NOTHING TO SHUT DOWN.
                   118
                   119         END

PUBLIC SYMBOLS
CCLEAR C 0011    ENDWE  C 002A    READ   C 0000    WECYCL C 0002

EXTERNAL SYMBOLS
WEDELY E 0000
USER SYMBOLS
CCLEAR C 0011    CLRACT A 0000    CLRCCL A 0023    CLRINA A 0003    CLRPRT A 0022    ENDWE  C 002A    READ   C 0000
WECYCL C 0002    WEDELY E 0000

ASSEMBLY COMPLETE,   NO ERRORS
```

AFN-01885A

Figure 23. Controller II Software Driver (Continued)

AFN-01913A

```
ASM80 :F1:CONT3.SRC MOD85
ISIS-II 8080/8085 MACRO ASSEMBLER, V3.0        MODULE   PAGE   1

 LOC  OBJ        LINE      SOURCE STATEMENT
                    1 $DEBUG
                    2
                    3
                    4         CSEG
                    5
                    6
                    7         PUBLIC  WECYCL,READ,ENDWE
                    8
                    9         EXTRN   WEDELY
                   10
                   11
                   12         ;       CONTROLLER III I/O PORT DEFINITIONS
                   13
                   14         ;       IMPLEMENTED IN 8155 RAM / I/O  / TIMER CHIP
                   15
                   16
                   17         ;       PORT    DESCRIPTION
                   18         ;       ----    -----------
                   19         ;       0A0H    PORT DIRECTION REGISTER (SET TO 0FH = ALL PORTS OUTPUT)
                   20         ;
                   21         ;       0A1H    2816 DATA (OUTPUT)
                   22         ;
                   23         ;       0A2H    2816 LOW ORDER ADDRESS, A0-A7 (OUTPUT)
                   24         ;
                   25         ;       0A3H    2816 HIGH ORDER ADDRESS AND CONTROL LINES (OUTPUT)
                   26         ;                       BITS 0-2:       A8-A10
                   27         ;                       BIT  3:         CE CTRL (0=SELECT READ,
                   28         ;                                                WRITE ENABLE)
                   29         ;                       BIT 4:          MUX CTRL (0=READ,1=WRITE)
                   30         ;                       BIT 5:          VPP CTRL (0=INACTIVE,1=ACTIVE)
                   31         ;
                   32         ;       0A4H    LOW ORDER TIMER COUNT REGISTER
                   33         ;
                   34         ;       0A5H    HIGH ORDER TIMER COUNT REGISTER
                   35         ;
                   36         ;       22H     CLEAR INTERRUPT FLIP-FLOPS PORT (OUTPUT)
                   37         ;                       BIT 0:          WRITE COMPL CLEAR (ACTIVE LOW)
                   38         ;                       BIT 1:          ILLEGAL ACC CLEAR (ACTIVE LOW)
                   39         ;                       BIT 5:          CHIP CLEAR MODE   (ACTIVE HI)
                   40
                   41
 00A0              42 EEPDR   EQU     0A0H            ; PORT DIRECTION REGISTER
 00A1              43 DATPRT  EQU     0A1H            ; 2816 DATA (OUTPUT)
 00A2              44 ADRPRT  EQU     0A2H            ; 2816 LOW ORDER ADDRESS (OUTPUT)
 00A3              45 CTLPRT  EQU     0A3H            ; 2816 HIGH ORDER ADDRESS AND CONTROL (OUTPUT)
 00A4              46 TIMLOW  EQU     0A4H            ; LOW ORDER TIMER COUNT REGISTER
 00A5              47 TIMHI   EQU     0A5H            ; HIGH ORDER TIMER COUNT REGISTER
                   48
 00C0              49 COUNTL  EQU     0C0H            ; LOW ORDER TIMER COUNT FOR 10 MSEC DELAY
 0083              50 COUNTH  EQU     83H             ; HIGH ORDER TIMER COUNT FOR 10 MSEC DELAY
```

AFN-01885A

Figure 24. Controller III Software Driver

AFN-01913A

```
ISIS-II 8080/8085 MACRO ASSEMBLER, V3.0        MODULE   PAGE   2
 LOC  OBJ        LINE       SOURCE STATEMENT

                   51
                   52
                   53         ;       CONTROLLER III READ SUBROUTINE
                   54
                   55         ;       DATA PASSED:    HL = ADDRESS OF 2816 LOCATION TO READ
                   56         ;       DATA RETURNED:  A  = DATA READ
                   57         ;       REGS DESTROYED: NONE
                   58
                   59 READ:
 0000 7E           60         MOV     A,M             ; JUST READ FROM MEMORY
 0001 C9           61         RET     ;
                   62
                   63
                   64
                   65
                   66         ;       CONTROLLER III WRITE/ERASE CYCLE SUBROUTINE
                   67
                   68         ;       DATA PASSED:    HL = ADDRESS OF 2816 LOCATION TO WRITE
                   69         ;                       A  = DATA TO WRITE
                   70         ;                                  OR 0FFH  (ERASE)
                   71         ;       DATA RETURNED:  NONE
                   72         ;       REGS DESTROYED: NONE
                   73         ;       RAM REQUIRED:   1 BYTE FOR TEMP ADDRESS/CONTROL STORAGE
                   74         ;       CALLS:          PEDELY (I/O POLL ROUTINE OR INTERRUPT DRIVER)
                   75         ;
                   76         ;       COMMENTS:       ENDWE (END OF WRITE/ERASE CYCLE) ROUTINE
                   77         ;                       IS CALLED BY INTERRUPT DRIVER OR I/O POLL
                   78         ;                       ROUTINE (WEDELY) TO SHUT DOWN CONTROLLER.
                   79         ;                       THIS SUBROUTINE IS PART OF THE DRIVER
                   80         ;                       PACKAGE ROUTINES INITIATED BY A CALL TO
                   81         ;                       PECYCL.
                   82
                   83
 0000              84 CLRACT  EQU     0H              ; ACTIVE CLEAR WRITE COMPL & ILL ACC FUNCTION
 0003              85 CLRINA  EQU     3H              ; INACTIVE CLEAR WC & IA FUNCTION
 0022              86 CLRPRT  EQU     22H             ; PORT USED TO CLEAR ILL ACC & WRT COMPL F/F
                   87
                   88
                   89 WECYCL:
 0002 F5           90         PUSH    PSW             ; SAVE REGISTERS
 0003 C5           91         PUSH    B
 0004 47           92         MOV     B,A             ; SAVE DATA TO WRITE IN B-REGISTER
 0005 3E00         93         MVI     A,CLRACT        ; CLEAR WRITE COMPLETE AND ILL ACC FLIP-FLOPS
 0007 D322         94         OUT     CLRPRT
 0009 3E03         95         MVI     A,CLRINA        ; DE-ACTIVATE CLEAR FUNCTION
 000B D322         96         OUT     CLRPRT
 000D 3E0F         97         MVI     A,0FH           ; PUT ALL 8155 I/O PORTS IN OUTPUT MODE
 000F D3A0         98         OUT     EEPDR           ; OUTPUT TO PORT DIRECTION REGISTER
 0011 78           99         MOV     A,B             ; FETCH DATA TO WRITE
 0012 D3A1        100         OUT     DATPRT          ; OUTPUT TO 2816 DATA LINES
 0014 7D          101         MOV     A,L             ; GET LOW ORDER ADDRES                    AFN-01885A
```

Figure 24. Controller III Software Driver (Continued)

AFN-01913A

```
ISIS-II 8080/8085 MACRO ASSEMBLER, V3.0        MODULE   PAGE   3
 LOC  OBJ          LINE        SOURCE STATEMENT

 0015 D3A2          102         OUT     ADRPRT          ; OUTPUT TO ADDRESS LINES
 0017 7C            103         MOV     A,H             ; GET HIGH ORDER ADDRESS
 0018 E607          104         ANI     7H              ; CLEAR ALL CONTROL LINES
 001A F610          105         ORI     10H             ; ADD MUX BIT TO SELECT I/O PORTS FOR WRITE
 001C D3A3          106         OUT     CTLPRT          ; OUTPUT HIGH ORDER ADDRESS AND CONTROL LINES
 001E F608          107         ORI     8H              ; ADD CE ACTIVE BIT
 0020 D3A3          108         OUT     CTLPRT          ; OUTPUT CONTROL LINES AGAIN
 0022 47            109         MOV     B,A             ; SAVE HIGH ORDER ADDR/CTL LINE DATA
 0023 3EC0          110         MVI     A,COUNT         ; OUTPUT TIMER COUNT (LOW ORDER)
 0025 D3A4          111         OUT     TIMLOW
 0027 3E83          112         MVI     A,COUNTH        ; OUTPUT TIMER COUNT (HIGH ORDER)
 0029 D3A5          113         OUT     TIMHI
 002B 3ECF          114         MVI     A,0CFH          ; START THE TIMER
 002D D3A0          115         OUT     EEPDR
 002F 78            116         MOV     A,B             ; RETRIEVE ADDRESS/CONTROL BITS
 0030 F620          117         ORI     20H             ; ADD VPP ACTIVE BIT
 0032 D3A3          118         OUT     CTLPRT          ; ACTIVATE VPP
 0034 320000    D   119         STA     TEMCTL          ; SAVE HIGH ADDRESS/CONTROL BITS FOR AFTER INTR
 0037 CD0000    E   120         CALL    WEDELY          ; WAIT FOR END OF WRITE CYCLE BY I/O POLL OR
                    121                                 ; INTERRUPT DRIVER ROUTINE
 003A C1            122         POP     B               ; RESTORE REGISTERS
 003B F1            123         POP     PSW
 003C C9            124         RET                     ; BACK TO CALLING ROUTINE
                    125
                    126
                    127
                    128         DSEG
                    129
 0000               130 TEMCTL: DS      1               ; RAM LOCATION FOR TEMP STORAGE OF CONTROL BITS
                    131
                    132
                    133
                    134         CSEG
                    135
                    136
                    137
                    138         ;       CONTROLLER III END-OF-WRITE/ERASE-CYCLE ROUTINE
                    139
                    140         ;       JUMPED TO BY I/O POLL OR INTERRUPT DRIVER AFTER WRITE COMPLETE
                    141         ;       TO SHUT DOWN CONTROLLER.
                    142
                    143         ;       DATA PASSED:    TEMCTL (1 RAM BYTE) CONTAINING HIGH ORDER
                    144         ;                       ADDRESS (3 BITS) & CONTROL BEFORE WRITE COMPL.
                    145
                    146 ENDWE:
 003D F5            147         PUSH    PSW             ; SAVE REGISTERS WE'LL DESTROY
 003E D5            148         PUSH    D
 003F 3A0000    D   149         LDA     TEMCTL          ; GET ADDRESS LINES/CONTROL BITS
 0042 E61F          150         ANI     1FH             ; REMOVE ACTIVE VPP BIT
 0044 D3A3          151         OUT     CTLPRT          ; DE-ACTIVATE VPP
                    152                                                          AFN-01885A
```

Figure 24. Controller III Software Driver (Continued)

AFN-01913A

```
ISIS-II 8080/8085 MACRO ASSEMBLER, V3.0        MODULE   PAGE   4
  LOC  OBJ        LINE       SOURCE STATEMENT

  0046 F5          153        PUSH    PSW            ; SAVE HIGH ORDER ADDRESS/CONTROL LINES
  0047 110D00      154        LXI     D,13D          ; SET UP COUNT FOR 100 USEC DELAY
                   155 DELAY:
  004A 1B          156        DCX     D              ; DELAY WHILE VPP FALLS
  004B 7A          157        MOV     A,D            ; DONE COUNTING?
  004C B3          158        ORA     E
  004D C24A00   C  159        JNZ     DELAY          ; NO: KEEP LOOPING
                   160
  0050 F1          161        POP     PSW            ; RESTORE ADDRESS/CONTROL LINES
  0051 E617        162        ANI     17H            ; REMOVE CE ACTIVE BIT
  0053 D3A3        163        OUT     CTLPRT         ; DE-ACTIVATE CE
  0055 E607        164        ANI     7H             ; REMOVE MUX SELECT WRITE BIT
  0057 D3A3        165        OUT     CTLPRT         ; LET MUX SELECT FOR READ OPERATIONS
  0059 3E0E        166        MVI     A,0EH          ; PUT DATA PORT BACK TO INPUT MODE
  005B D3A0        167        OUT     EEPDR          ; SO AS NOT TO CAUSE CONTENTION W/ DATA BUS
                   168
  005D D1          169        POP     D              ; RESTORE REGISTERS
  005E F1          170        POP     PSW
  005F C9          171        RET                    ; AND EXIT
                   172
                   173
                   174
                   175        END


PUBLIC SYMBOLS
ENDWE  C 003D    READ   C 0000    WECYCL C 0002

EXTERNAL SYMBOLS
WEDELY E 0000
USER SYMBOLS
ADRPRT A 00A2    CLRACT A 0000    CLRINA A 0003    CLRPRT A 0022    COUNTH A 0083    COUNTL A 00C0    CTLPRT A 00A3
DATPRT A 00A1    DELAY  C 004A    EEPDR  A 00A0    ENDWE  C 003D    READ   C 0000    TEMCTL D 0000    TIMHI  A 00A5
TIMLOW A 00A4    WECYCL C 0002    WEDELY E 0000

ASSEMBLY COMPLETE,   NO ERRORS                                                          AFN-01885A
```

Figure 24. Controller III Software Driver (Continued)

AFN-01913A

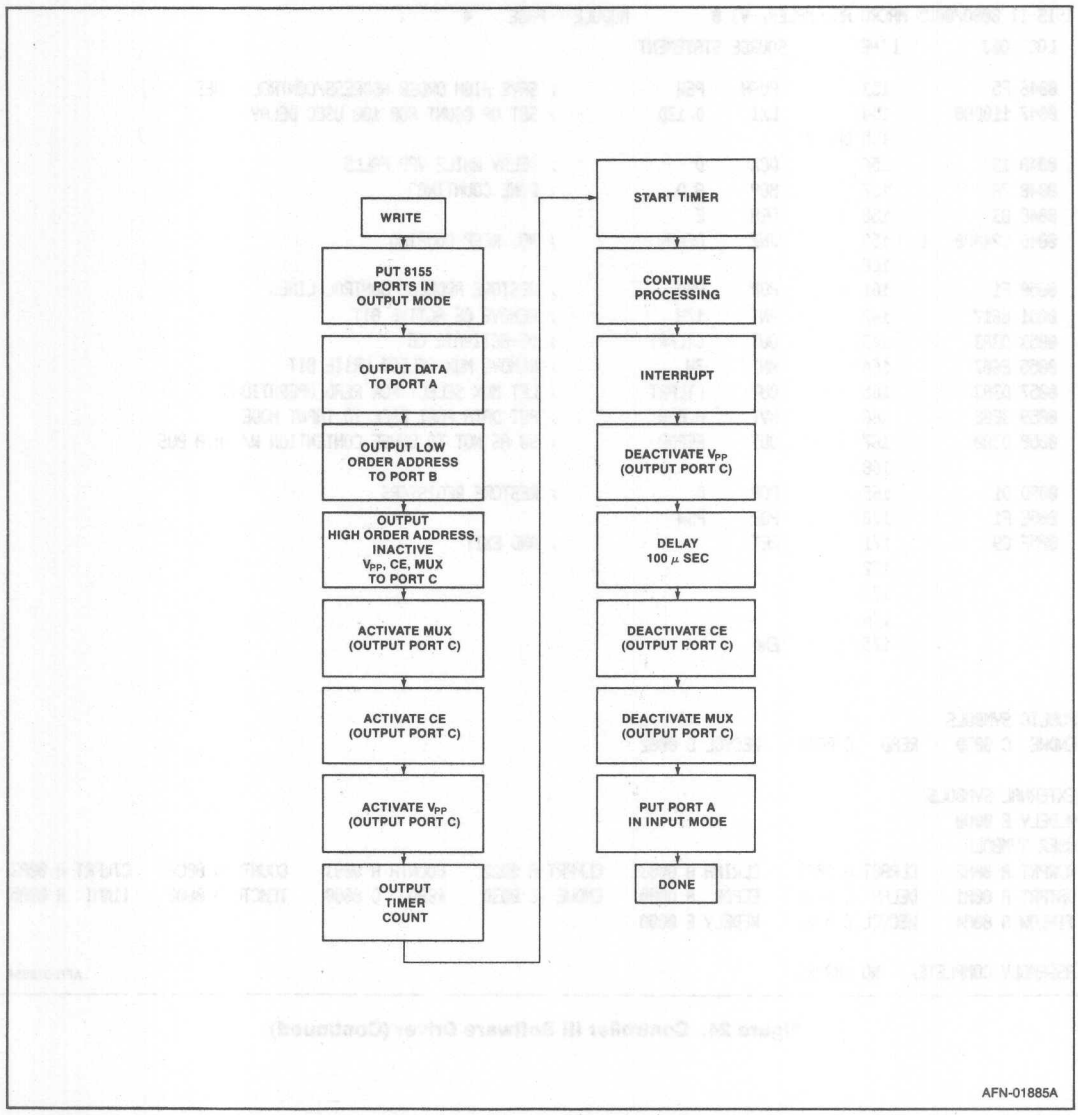


Figure 24A. Controller III, IV, Flowchart

AFN-01913A

```
ASM80 :F1:CONT4.SRC MOD85
ISIS-II 8080/8085 MACRO ASSEMBLER, V3.0        MODULE   PAGE    1

  LOC  OBJ        LINE        SOURCE STATEMENT
                     1 $DEBUG
                     2
                     3
                     4         PUBLIC  WECYCL,READ,ENDWE
                     5
                     6         EXTRN   WEDELY
                     7
                     8         CSEG
                     9
                    10
                    11         ;       CONTROLLER IV I/O PORT DEFINITIONS
                    12
                    13         ;       IMPLEMENTED IN 8155 RAM / I/O  / TIMER CHIP
                    14
                    15
                    16         ;       PORT    DESCRIPTION
                    17         ;       ----    -----------
                    18         ;       0A0H    PORT DIRECTION REGISTER (SET TO 0FH = ALL PORTS OUTPUT)
                    19         ;
                    20         ;       0A1H    2816 DATA (OUTPUT)
                    21         ;
                    22         ;       0A2H    2816 LOW ORDER ADDRESS, A0-A7 (OUTPUT)
                    23         ;
                    24         ;       0A3H    2816 HIGH ORDER ADDRESS AND CONTROL LINES (OUTPUT)
                    25         ;                       BITS 0-2:       A8-A10
                    26         ;                       BIT  3:         CE CTRL (0=SELECT READ,
                    27         ;                                               WRITE ENABLE)
                    28         ;                       BIT 4:          MUX CTRL (0=READ,1=WRITE)
                    29         ;                       BIT 5:          VPP CTRL (0=INACTIVE,1=ACTIVE)
                    30         ;
                    31         ;       0A4H    LOW ORDER TIMER COUNT REGISTER
                    32         ;
                    33         ;       0A5H    HIGH ORDER TIMER COUNT REGISTER
                    34         ;
                    35         ;       22H     PORT USED TO CLEAR WRITE COMPL & ILLEGAL ACC INTERRUPTS
                    36
                    37
  00A0              38 EEPDR   EQU     0A0H            ; PORT DIRECTION REGISTER
  00A1              39 DATPRT  EQU     0A1H            ; 2816 DATA (OUTPUT)
  00A2              40 ADRPRT  EQU     0A2H            ; 2816 LOW ORDER ADDRESS (OUTPUT)
  00A3              41 CTLPRT  EQU     0A3H            ; 2816 HIGH ORDER ADDRESS AND CONTROL (OUTPUT)
  00A4              42 TIMLOW  EQU     0A4H            ; LOW ORDER TIMER COUNT REGISTER
  00A5              43 TIMHI   EQU     0A5H            ; HIGH ORDER TIMER COUNT REGISTER
                    44
  00C0              45 COUNTL  EQU     0C0H            ; LOW ORDER TIMER COUNT FOR 10 MSEC DELAY
  0083              46 COUNTH  EQU     83H             ; HIGH ORDER TIMER COUNT FOR 10 MSEC DELAY
                    47
                    48
                    49         ;       CONTROLLER IV READ SUBROUTINE
                    50
```

AFN-01885A

Figure 25. Controller IV Software Driver

AFN-01913A

```
ISIS-II 8080/8085 MACRO ASSEMBLER, V3.0        MODULE   PAGE    2
  LOC  OBJ         LINE        SOURCE STATEMENT

                     51        ;       DATA PASSED:    HL = ADDRESS OF 2816 LOCATION TO READ
                     52        ;       DATA RETURNED:  A  = DATA READ
                     53        ;       REGS DESTROYED: FLAGS
                     54
                     55 READ:
  0000 3E0E          56        MVI     A,0EH           ; PUT DATA PRT IN INPUT MODE, ALL OTHERS-OUTPUT
  0002 D3A0          57        OUT     EEPDR           ; OUTPUT TO PORT DIRECTRION REGISTER
  0004 7D            58        MOV     A,L             ; GET LOW ORDER ADDRESS
  0005 D3A2          59        OUT     ADRPRT          ; OUTPUT TO ADDRESS PORT
  0007 7C            60        MOV     A,H             ; GET HIGH ORDER ADDRESS
  0008 E607          61        ANI     7H              ; REMOVE ALL CONTROL BITS (KEEP 3 BIT ADDRESS)
  000A F610          62        ORI     10H             ; ADD OE' INACTIVE BIT
  000C D3A3          63        OUT     CTLPRT          ; OUTPUT TO CONTROL PORT
  000E E607          64        ANI     07H             ; REMOVE OE' INACTIVE BIT (ACTIVATE OE)
  0010 F608          65        ORI     8H              ; ADD CE' ACTIVE BIT
  0012 D3A3          66        OUT     CTLPRT          ; OUTPUT TO CONTROL PORT
  0014 DBA1          67        IN      DATPRT          ; INPUT DATA FROM 2816
  0016 F5            68        PUSH    PSW             ; SAVE DATA
  0017 AF            69        XRA     A               ; ZERO A REGISTER
  0018 D3A3          70        OUT     CTLPRT          ; DEACTIVATE ALL CONTROL LINES
  001A F1            71        POP     PSW             ; RESTORE DATA
  001B C9            72        RET                     ; AND EXIT
                     73
                     74
                     75
                     76
                     77        ;       CONTROLLER IV WRITE/ERASE CYCLE SUBROUTINE
                     78
                     79        ;       DATA PASSED:    HL = ADDRESS OF 2816 LOCATION TO WRITE
                     80        ;                       A  = DATA TO WRITE
                     81        ;                            OR 0FFH  (ERASE)
                     82        ;       DATA RETURNED:  NONE
                     83        ;       REGS DESTROYED: NONE
                     84        ;       RAM REQUIRED:   1 BYTE FOR TEMP ADDRESS/CONTROL STORAGE
                     85        ;       CALLS:          WEDELY (I/O POLL ROUTINE OR INTERRUPT DRIVER)
                     86        ;
                     87        ;       COMMENTS:       ENDWE (END OF WRITE/ERASE CYCLE) ROUTINE
                     88        ;                       IS CALLED BY INTERRUPT DRIVER OR I/O POLL
                     89        ;                       ROUTINE (WEDELY) TO SHUT DOWN CONTROLLER.
                     90        ;                       THIS SUBROUTINE IS PART OF THE DRIVER
                     91        ;                       PACKAGE ROUTINES INITIATED BY A CALL TO
                     92        ;                       WECYCL.
                     93
                     94
  0022               95 CLRPRT  EQU     22H             ; I/O PORT USED TO RESET INTERRUPT F/F'S
  0000               96 CLRACT  EQU     0H              ; ACTIVATE CLEAR WRITE COMPL & ILL ACC INTR
  0003               97 CLRINA  EQU     3H              ; INACTIVE CLEAR WC & IA FUNCTION
                     98
                     99
                    100 WECYCL.
  001C F5           101        PUSH    PSW             ; SAVE REGISTERS                    AFN-01885A
```

Figure 25. Controller IV Software Driver (Continued)

AFN-01913A

```
ISIS-II 8080/8085 MACRO ASSEMBLER, V3.0        MODULE   PAGE   3
 LOC  OBJ          LINE        SOURCE STATEMENT

 001D C5           102         PUSH    B
 001E 47           103         MOV     B,A             ; SAVE DATA TO WRITE IN B-REGISTER
 001F 3E00         104         MVI     A,CLRACT        ; CLEAR WRITE COMPLETE & ILLEGAL ACCESS F/F'S
 0021 D322         105         OUT     CLRPRT          ; ACTIVATE CLEAR FUNCTION
 0023 3E03         106         MVI     A,CLRINA        ; DEACTIVATE CLEAR FUNCTION
 0025 D322         107         OUT     CLRPRT
 0027 3E0F         108         MVI     A,0FH           ; PUT ALL 8155 I/O PORTS IN OUTPUT MODE
 0029 D3A0         109         OUT     EEPDR           ; OUTPUT TO PORT DIRECTION REGISTER
 002B 78           110         MOV     A,B             ; FETCH DATA TO WRITE
 002C D3A1         111         OUT     DATPRT          ; OUTPUT TO 2816 DATA LINES
 002E 7D           112         MOV     A,L             ; GET LOW ORDER ADDRES
 002F D3A2         113         OUT     ADRPRT          ; OUTPUT TO ADDRESS LINES
 0031 7C           114         MOV     A,H             ; GET HIGH ORDER ADDRESS
 0032 E607         115         ANI     7H              ; CLEAR ALL CONTROL LINES
 0034 F610         116         ORI     10H             ; ADD MUX BIT TO SELECT I/O PORTS FOR WRITE
 0036 D3A3         117         OUT     CTLPRT          ; OUTPUT HIGH ORDER ADDRESS AND CONTROL LINES
 0038 F608         118         ORI     8H              ; ADD CE ACTIVE BIT
 003A D3A3         119         OUT     CTLPRT          ; OUTPUT CONTROL LINES AGAIN
 003C 47           120         MOV     B,A             ; SAVE HIGH ORDER ADDR/CTL LINE DATA
 003D 3EC0         121         MVI     A,COUNTL        ; OUTPUT TIMER COUNT (LOW ORDER)
 003F D3A4         122         OUT     TIMLOW
 0041 3E83         123         MVI     A,COUNTH        ; OUTPUT TIMER COUNT (HIGH ORDER)
 0043 D3A5         124         OUT     TIMHI
 0045 3ECF         125         MVI     A,0CFH          ; START THE TIMER
 0047 D3A0         126         OUT     EEPDR
 0049 78           127         MOV     A,B             ; RETRIEVE ADDRESS/CONTROL BITS
 004A F620         128         ORI     20H             ; ADD VPP ACTIVE BIT
 004C D3A3         129         OUT     CTLPRT          ; ACTIVATE VPP
 004E 320000   D   130         STA     TEMCTL          ; SAVE HIGH ADDRESS/CONTROL BITS FOR AFTER INTR
 0051 C1           131         POP     B               ; RESTORE REGISTERS
 0052 F1           132         POP     PSW
 0053 CD0000   E   133         CALL    WEDELY          ; GO TO I/O POLL LOOP OR INTERRUPT DRIVER
 0056 C9           134         RET                     ; AND RETURN BACK TO MAIN PROGRAM
                   135
                   136
                   137
                   138         DSEG
                   139
 0000              140 TEMCTL: DS      1               ; RAM LOCATION FOR TEMP STORAGE OF CONTROL BITS
                   141
                   142
                   143
                   144         CSEG
                   145
                   146
                   147         ;       CONTROLLER IV END-OF-WRITE/ERASE-CYCLE ROUTINE
                   148
                   149         ;       CALLED TO BY I/O POLL OR INTERRUPT DRIVER AFTER WRITE COMPLETE
                   150         ;       TO SHUT DOWN CONTROLLER.
                   151
```

AFN-01885A

Figure 25. Controller IV Software Driver (Continued)

AFN-01913A

```
ISIS-II 8080/8085 MACRO ASSEMBLER, V3.0        MODULE   PAGE   4
  LOC  OBJ         LINE       SOURCE STATEMENT

                    152        ;       DATA PASSED:    TEMCTL (1 RAM BYTE) CONTAINING HIGH ORDER
                    153        ;                       ADDRESS (3 BITS) & CONTROL BEFORE WRITE COMPL.
                    154
                    155 ENDWE:
  0057 F5           156        PUSH    PSW             ; SAVE REGISTERS WE'LL DESTROY
  0058 D5           157        PUSH    D
  0059 3A0000   D   158        LDA     TEMCTL          ; GET ADDRESS LINES/CONTROL BITS
  005C E61F         159        ANI     1FH             ; REMOVE ACTIVE VPP BIT
  005E D3A3         160        OUT     CTLPRT          ; DE-ACTIVATE VPP
                    161
  0060 F5           162        PUSH    PSW             ; SAVE HIGH ORDER ADDRESS/CONTROL LINES
  0061 110D00       163        LXI     D,13D           ; SET UP COUNT FOR 100 USEC DELAY
                    164 DELAY:
  0064 1B           165        DCX     D               ; DELAY WHILE VPP FALLS
  0065 7A           166        MOV     A,D             ; DONE COUNTING?
  0066 B3           167        ORA     E
  0067 C26400   C   168        JNZ     DELAY           ; NO; KEEP LOOPING
                    169
  006A F1           170        POP     PSW             ; RESTORE ADDRESS/CONTROL LINES
  006B E617         171        ANI     17H             ; REMOVE CE ACTIVE BIT
  006D D3A3         172        OUT     CTLPRT          ; DE-ACTIVATE CE
  006F E607         173        ANI     7H              ; REMOVE MUX SELECT WRITE BIT
  0071 D3A3         174        OUT     CTLPRT          ; LET MUX SELECT FOR READ OPERATIONS
  0073 3E0E         175        MVI     A,0EH           ; PUT DATA PORT BACK TO INPUT MODE
  0075 D3A0         176        OUT     EEPDR           ; SO AS NOT TO CAUSE CONTENTION W/ DATA BUS
                    177
  0077 D1           178        POP     D               ; RESTORE REGISTERS
  0078 F1           179        POP     PSW
  0079 C9           180        RET                     ; AND EXIT
                    181
                    182
                    183
                    184        END


PUBLIC SYMBOLS
ENDWE  C 0057    READ   C 0000    WECYCL C 001C

EXTERNAL SYMBOLS
WEDELY E 0000

USER SYMBOLS
ADRPRT A 00A2    CLRACT A 0000    CLRINA A 0003    CLRPRT A 0022    COUNTH A 0083    COUNTL A 00C0    CTLPRT A 00A3
DATPRT A 00A1    DELAY  C 0064    EEPDR  A 00A0    ENDWE  C 0057    READ   C 0000    TEMCTL D 0000    TIMHI  A 00A5
TIMLOW A 00A4    WECYCL C 001C    WEDELY E 0000

ASSEMBLY COMPLETE,  NO ERRORS                                                          AFN-01885A
```

Figure 25. Controller IV Software Driver (Continued)

AFN-01913A

```
ASM80 :F1:CCLR2.SRC MOD85
ISIS-II 8080/8085 MACRO ASSEMBLER, V3.0        MODULE   PAGE   1

 LOC  OBJ        LINE     SOURCE STATEMENT

                    1 $DEBUG
                    2
                    3
                    4         PUBLIC  CERASE
                    5
                    6         EXTRN   WEDELY,ENDWE
                    7
                    8         CSEG
                    9
                   10
                   11
                   12         ;       CONTROLLER II CHIP ERASE SUBROUTINE
                   13
                   14         ;       DATA PASSED:    NONE
                   15         ;       DATA RETURNED:  NONE
                   16         ;       REGS DESTROYED: NONE
                   17         ;       CALLS           WEDELY ( I/O POLL ROUTINE OR INTERRUPT DRIVER)
                   18
                   19         ;       I/O PORTS USED:
                   20         ;               PORT 22H (OUTPUT)
                   21         ;                       BIT 0 = WRITE COMPLETE CLEAR (ACTIVE LOW)
                   22         ;                       BIT 1 = ILLEGAL ACCESS CLEAR (ACTIVE LOW)
                   23         ;                       BIT 5 = CHIP CLR (+12V TO OE' LINE) (ACTIVE HI)
                   24
                   25         ;       COMMENTS:       ENDWE (END OF WRITE/ERASE CYCLE) ROUTINE
                   26         ;                       IS CALLED BY INTERRUPT DRIVER OR I/O POLL
                   27         ;                       ROUTINE (WEDELY) TO SHUT DOWN CONTROLLER.
                   28         ;                       THIS SUBROUTINE IS PART OF THE DRIVER
                   29         ;                       PACKAGE ROUTINES INITIATED BY A CALL TO
                   30         ;                       CERASE.
                   31
                   32
                   33         ;       I/O SYMBOLS
                   34
  0022             35 CLRPRT  EQU     22H             ; CHIP ERASE OUTPUT PORT
  0000             36 CLRACT  EQU     00H             ; ACTIVE RESET OF CLEAR WC & ILL ACC FLIP-FLOPS
  0003             37 CLRINA  EQU     03H             ; INACTIVE RESET OF CLEAR WC & IA FUNCTION
  0023             38 CLRCCL  EQU     23H             ; DATA TO DEACTIVATE CLEAR WC & IA BUT ACTIVATE
                   39                                 ; OE' = +12V FUNCTION FOR CHIP CLEAR
                   40 CERASE:
  0000 F5          41         PUSH    PSW             ; SAVE REGISTERS
  0001 E5          42         PUSH    H
  0002 3E00        43         MVI     A,CLRACT        ; GET BITS TO RESET WRITE COMPL AND ILL ACC
  0004 D322        44         OUT     CLRPRT
  0006 3E23        45         MVI     A,CLRCCL        ; GET BITS TO DEACTIVATE CLEAR FUNCTION AND
                   46                                 ; TURN ON OE' = +12V FUNCTION FOR CHIP ERASE
  0008 D322        47         OUT     CLRPRT          ; OUTPUT TO I/O PORT
  000A 3EFF        48         MVI     A,0FFH          ; WRITE 0FFH TO THE 2816
  000C 3200A0      49         STA     0A000H                                          AFN-01885A
```

Figure 26. Controller II Chip Erase Routines

AFN-01913A

```
ISIS-II 8080/8085 MACRO ASSEMBLER, V3.0        MODULE   PAGE   2
  LOC  OBJ        LINE       SOURCE STATEMENT

  000F CD0000   E    50        CALL    WEDELY          ; GO TO I/O POLL LOOP OR INTERRUPT DRIVER
  0012 3E03          51        MVI     A,CLRINA        ; DEACTIVATE CHIP CLEAR FUNCTION
  0014 D322          52        OUT     CLRPRT
  0016 E1            53        POP     H               ; RESTORE REGISTERS
  0017 F1            54        POP     PSW
  0018 C9            55        RET
                     56
                     57        END


PUBLIC SYMBOLS
CERASE C 0000

EXTERNAL SYMBOLS
ENDWE  E 0000    WEDELY E 0000

USER SYMBOLS
CERASE C 0000    CLRACT A 0000    CLRCCL A 0023    CLRINA A 0003    CLRPRT A 0022    ENDWE  E 0000    WEDELY E 0000

ASSEMBLY COMPLETE,   NO ERRORS                                                                    AFN-01885A
```

Figure 26. Controller II Chip Erase Routines (Continued)

AFN-01913A

```
ASM80 :F1:CCLR34.SRC MOD85
ISIS-II 8080/8085 MACRO ASSEMBLER, V3.0        MODULE   PAGE   1

 LOC  OBJ        LINE      SOURCE STATEMENT
                    1 $DEBUG
                    2
                    3
                    4         CSEG
                    5
                    6
                    7         PUBLIC  CERASE
                    8
                    9         EXTRN   WEDELY,ENDWE
                   10
                   11
                   12         ;       CONTROLLER III I/O PORT DEFINITIONS
                   13
                   14         ;       IMPLEMENTED IN 8155 RAM / I/O  / TIMER CHIP
                   15
                   16
                   17         ;       PORT    DESCRIPTION
                   18         ;       ----    -----------
                   19         ;       0A0H    PORT DIRECTION REGISTER (SET TO 0FH = ALL PORTS OUTPUT)
                   20         ;
                   21         ;       0A1H    2816 DATA (OUTPUT)
                   22         ;
                   23         ;       0A2H    2816 LOW ORDER ADDRESS, A0-A7 (OUTPUT)
                   24         ;
                   25         ;       0A3H    2816 HIGH ORDER ADDRESS AND CONTROL LINES (OUTPUT)
                   26         ;                       BITS 0-2:       A8-A10
                   27         ;                       BIT  3:         CE CTRL (0=SELECT READ,
                   28         ;                                               WRITE ENABLE)
                   29         ;                       BIT 4:          MUX CTRL (0=READ,1=WRITE)
                   30         ;                       BIT 5:          VPP CTRL (0=INACTIVE,1=ACTIVE)
                   31         ;
                   32         ;       0A4H    LOW ORDER TIMER COUNT REGISTER
                   33         ;
                   34         ;       0A5H    HIGH ORDER TIMER COUNT REGISTER
                   35         ;
                   36         ;       22H     CHIP ERASE, INTERRUPT F/F CLEAR PORTS (OUTPUT)
                   37         ;                       BIT 0:          WRITE COMPL CLEAR (ACTIVE LOW)
                   38         ;                       BIT 1:          ILLEGAL ACC CLEAR (ACTIVE LOW)
                   39         ;                       BIT 5:          CHIP ERASE (+12V TO OE') ACT HI
                   40
                   41
                   42
 00A0              43 EEPDR   EQU     0A0H            ; PORT DIRECTION REGISTER
 00A1              44 DATPRT  EQU     0A1H            ; 2816 DATA (OUTPUT)
 00A2              45 ADRPRT  EQU     0A2H            ; 2816 LOW ORDER ADDRESS (OUTPUT)
 00A3              46 CTLPRT  EQU     0A3H            ; 2816 HIGH ORDER ADDRESS AND CONTROL (OUTPUT)
 00A4              47 TIMLOW  EQU     0A4H            ; LOW ORDER TIMER COUNT REGISTER
 00A5              48 TIMHI   EQU     0A5H            ; HIGH ORDER TIMER COUNT REGISTER
                   49
                                                                                        AFN-01885A
```

Figure 27. Controller III, IV Chip Erase Routines

```
ISIS-II 8080/8085 MACRO ASSEMBLER, V3.0        MODULE   PAGE    2
 LOC  OBJ        LINE       SOURCE STATEMENT

 00C0             50 COUNTL  EQU     0C0H           ; LOW ORDER TIMER COUNT FOR 10 MSEC DELAY
 0083             51 COUNTH  EQU     83H            ; HIGH ORDER TIMER COUNT FOR 10 MSEC DELAY
                  52
                  53
                  54
                  55         ;       CONTROLLER III, IV CHIP ERASE SUBROUTINE
                  56
                  57         ;       DATA PASSED:    HL = ADDRESS OF 2816 LOCATION TO WRITE
                  58         ;                       A  = DATA TO WRITE
                  59         ;                            OR 0FFH  (ERASE)
                  60         ;       DATA RETURNED:  NONE
                  61         ;       REGS DESTROYED: NONE
                  62         ;       RAM REQUIRED:   1 BYTE FOR TEMP ADDRESS/CONTROL STORAGE
                  63         ;       CALLS:          PEDELY (I/O POLL ROUTINE OR INTERRUPT DRIVER)
                  64         ;
                  65         ;       COMMENTS:       ENDWE (END OF WRITE/ERASE CYCLE) ROUTINE
                  66         ;                       IS CALLED BY INTERRUPT DRIVER OR I/O POLL
                  67         ;                       ROUTINE (WEDELY) TO SHUT DOWN CONTROLLER.
                  68         ;                       THIS SUBROUTINE IS PART OF THE DRIVER
                  69         ;                       PACKAGE ROUTINES INITIATED BY A CALL TO
                  70         ;                       WECYCL.
                  71
                  72
 0000             73 CLRACT  EQU     0H             ; ACTIVE CLEAR WRITE COMPL & ILL ACC FUNCTION
 0023             74 CLRCCL  EQU     23H            ; DATA TO DEACTIVATE CLEAR WC & IA BUT ACTIVATE
                  75                                ; OE' = +12V FUNCTION FOR CHIP ERASE
 0003             76 CLRINA  EQU     3H             ; INACTIVE CLEAR WC & IA FUNCTION
 0022             77 CLRPRT  EQU     22H            ; PORT USED TO CLEAR ILL ACC & WRT COMPL F/F
                  78
                  79
                  80 CERASE:
 0000 F5          81         PUSH    PSW            ; SAVE REGISTERS
 0001 3E00        82         MVI     A,CLRACT       ; CLEAR WRITE COMPLETE AND ILL ACC FLIP-FLOPS
 0003 D322        83         OUT     CLRPRT
 0005 3E23        84         MVI     A,CLRCCL       ; DE-ACTIVATE CLEAR FUNCTION & SET OE' = +12V
 0007 D322        85         OUT     CLRPRT
 0009 3E0F        86         MVI     A,0FH          ; PUT ALL 8155 I/O PORTS IN OUTPUT MODE
 000B D3A0        87         OUT     EEPDR          ; OUTPUT TO PORT DIRECTION REGISTER
 000D 3EFF        88         MVI     A,0FFH         ; DATA TO WRITE IS ALL 1'S
 000F D3A1        89         OUT     DATPRT         ; OUTPUT TO 2816 DATA LINES
 0011 3E00        90         MVI     A,0            ; LOW ORDER ADDR (WE WRITE TO A000 FOR CCLR)
 0013 D3A2        91         OUT     ADRPRT         ; OUTPUT TO ADDRESS LINES
 0015 3E10        92         MVI     A,10H          ; ACTIVATE MUX FOR WRITE OPERATION
 0017 D3A3        93         OUT     CTLPRT         ; OUTPUT HIGH ORDER ADDRESS AND CONTROL LINES
 0019 F608        94         ORI     8H             ; ADD CE ACTIVE BIT
 001B D3A3        95         OUT     CTLPRT         ; OUTPUT CONTROL LINES AGAIN
 001D 3EC0        96         MVI     A,COUNTL       ; OUTPUT TIMER COUNT (LOW ORDER)
 001F D3A4        97         OUT     TIMLOW
 0021 3E83        98         MVI     A,COUNTH       ; OUTPUT TIMER COUNT (HIGH ORDER)
 0023 D3A5        99         OUT     TIMHI
 0025 3ECF       100         MVI     A,0CFH         ; START THE TIMER                    AFN-01885A
```

Figure 27. Controller III, IV Chip Erase Routines (Continued)

AFN-01913A

```
ISIS-II 8080/8085 MACRO ASSEMBLER, V3.0        MODULE   PAGE    3
  LOC  OBJ         LINE        SOURCE STATEMENT

  0027 D3A0         101         OUT     EEPDR
  0029 3E38         102         MVI     A,38H           ; ACTIVATE VPP, CE' AND MUX
  002B D3A3         103         OUT     CTLPRT          ; ACTIVATE VPP
  002D 320000   D   104         STA     TEMCTL          ; SAVE HIGH ADDRESS/CONTROL BITS FOR AFTER INTR
  0030 CD0000   E   105         CALL    WEDELY          ; WAIT FOR END OF WRITE CYCLE BY I/O POLL OR
                    106                                 ; INTERRUPT DRIVER ROUTINE
  0033 3E03         107         MVI     A,CLRINA        ; DEACTIVATE CHIP CLEAR FUNCTION
  0035 D322         108         OUT     CLRPRT
  0037 F1           109         POP     PSW
  0038 C9           110         RET                     ; BACK TO CALLING ROUTINE
                    111
                    112
                    113
                    114         DSEG
                    115
  0000              116 TEMCTL: DS      1               ; RAM LOCATION FOR TEMP STORAGE OF CONTROL BITS
                    117
                    118
                    119
                    120         END


PUBLIC SYMBOLS
CERASE C 0000

EXTERNAL SYMBOLS
ENDWE  E 0000    WEDELY E 0000

USER SYMBOLS
ADRPRT A 00A2    CERASE C 0000    CLRACT A 0000    CLRCCL A 0023    CLRINA A 0003    CLRPRT A 0022    COUNTH A 0083
COUNTL A 00C0    CTLPRT A 00A3    DATPRT A 00A1    EEPDR  A 00A0    ENDWE  E 0000    TEMCTL D 0000    TIMHI  A 00A5
TIMLOW A 00A4    WEDELY E 0000

ASSEMBLY COMPLETE,   NO ERRORS                                                              AFN-01885A
```

Figure 27. Controller III, IV Chip Erase Routines (Continued)

AFN-01913A

```
ASM80 :F1:IOPOLL.SRC
ISIS-II 8080/8085 MACRO ASSEMBLER, V3.0        MODULE   PAGE    1


 LOC  OBJ        LINE       SOURCE STATEMENT

                    1 $DEBUG
                    2
                    3
                    4
                    5
                    6        ;       ****************************************************
                    7
                    8        ;               2816 CONTROLLER I/O POLL ROUTINE
                    9
                   10        ;
                   11        ;
                   12
                   13        ;       ****************************************************
                   14
                   15
                   16        PUBLIC  PEDELY
                   17
                   18        EXTRN   ENDPE
                   19
                   20
                   21        CSEG
                   22
                   23
                   24        ;       PEDELY:  PROGRAM/ERASE CYCLE DELAY ROUTINE
                   25
                   26        ;                DELAYS VIA I/O POLLED WAIT LOOP ON 'WRITE COMPLETE'
                   27        ;                BIT.
                   28
                   29        ;       DATA PASSED:    NONE
                   30        ;       DATA RETURNED:  NONE
                   31        ;       REGS DESTROYED: NONE
                   32
                   33        ;       I/O PORT USED:  PORT 21H
                   34        ;                         - BIT 1 = 'WRITE COMPLETE' (ACTIVE HIGH)
                   35
                   36
 0021              37 WCPORT  EQU     21H             ; I/O PORT CONTAINING WRITE COMPLETE BIT
                   38
                   39
                   40 PEDELY:
 0000 F5           41         PUSH    PSW             ; SAVE A-REG, FLAGS
                   42 LOOP:
 0001 DB21         43         IN      WCPORT          ; GET WRITE COMPLETE BIT
 0003 E602         44         ANI     2H              ; MASK WC BIT
 0005 CA0100   C   45         JZ      LOOP            ; IF NOT SET THEN KEEP WAITING
                   46
 0008 F1           47         POP     PSW             ; RESTORE A, FLAGS                    AFN-01885A
```

Figure 28. Controller I/O Poll Routines

AFN-01913A

```
ISIS-II 8080/8085 MACRO ASSEMBLER, V3.0        MODULE   PAGE    2
  LOC             LINE        SOURCE STATEMENT

  0009 CD0000   E    48         CALL    ENDPE          ; CALL END PROGRAM/ERASE CYCLE ROUTINE TO
                     49                                ; SHUT DOWN 2816.
  000C C9            50         RET                    ; RETURN BACK TO HOST PROGRAM.
                     51
                     52
                     53
                     54         END


PUBLIC SYMBOLS
PEDELY C 0000

EXTERNAL SYMBOLS
ENDPE  E 0000

USER SYMBOLS
ENDPE  E 0000    LOOP   C 0001    PEDELY C 0000    WCPORT A 0021

ASSEMBLY COMPLETE,  NO ERRORS                                          AFN-01885A
```

Figure 28. Controller I/O Poll Routines (Continued)

```
ASM80 :F1:INTER.SRC MOD85
ISIS-II 8080/8085 MACRO ASSEMBLER, V3.0        MODULE   PAGE   1

 LOC  OBJ        LINE      SOURCE STATEMENT
                    1 $DEBUG
                    2
                    3
                    4
                    5         CSEG
                    6
                    7         PUBLIC  WEDELY,HANDLE
                    8
                    9         EXTRN   ENDWE
                   10
                   11         CSEG
                   12
                   13
                   14
                   15         ;       WEDELY - WRITE/ERASE CYCLE DELAY SUBROUTINE
                   16         ;              - INTERRUPT DRIVEN
                   17
                   18         ;       CALLED TO WAIT FOR INTERRUPT TO OCCUR WHILE WAITING OUT
                   19         ;       2816 CONTROLLER WRITE CYCLE
                   20
                   21         ;       DATA PASSED:    NONE
                   22         ;       REGS DESTROYED: NONE
                   23         ;       INTERRUPT USED: EXPECTS CONTROLLER TO USE INTERRUPT 6.5
                   24         ;                       MASKS OUT ALL OTHER INTERRUPTS
                   25         ;       USED WITH:      INTERRUPT HANDLER SUBOURINE 'HANDLE'
                   26
                   27         ;       RAM REQD:       1 BYTE - 'WRTCOM' - WRITE COMPLETE INTERCOM
                   28         ;                              - BIT ZERO SET BY INTERRUPT HANDLER
                   29         ;                                WHEN WRITE COMPLETE.
                   30
                   31
  000D             32 IONMSK  EQU     1101B          ; INTERRUPT MASK ENABLING INTERRUPT 6.5 ONLY
                   33
                   34 WEDELY:
  0000 F5          35         PUSH    PSW            ; SAVE A-REGISTER, FLAGS
  0001 AF          36         XRA     A              ; ZERO WRITE COMPLETE INTERCOM REGISTER
  0002 320000    D 37         STA     WRTCOM
  0005 3E0D        38         MVI     A,IONMSK       ; ENABLE INTERRUPT 6.5 ONLY
  0007 30          39         SIM
  0008 FB          40         EI                     ; ALLOW INTERRUPTS TO OCCUR
                   41 LOOP:
  0009 3A0000    D 42         LDA     WRTCOM         ; GET WRITE COMPLETE STATUS REGISTER
  000C 1F          43         RAR                    ; PUT LEAST SIGNIFICANT BIT INTO CARRY
  000D D20900    C 44         JNC     LOOP           ; IF LSB NOT SET, KEEP LOOPING
                   45
  0010 F1          46         POP     PSW            ; RESTORE A-REGISTER
  0011 C9          47         RET                    ; BACK TO HOST PROGRAM
                   48
                   49
                   50
```

AFN-01885A

Figure 29. Controller Interrupt Driver

AFN-01913A

```
ISIS-II 8080/8085 MACRO ASSEMBLER, V3.0        MODULE  PAGE   2
 LOC  OBJ        LINE       SOURCE STATEMENT

                   51
                   52          DSEG                ; SAVE A RAM LOCATION
 0000              53 WRTCOM:  DS      1
                   54
                   55
                   56
                   57
                   58          ASEG
                   59
 0FE0              60          ORG     0FE0H
                   61
                   62
                   63
                   64          ;       HANDLE - 2816 CONTROLLER INTERRUPT HANDLER
                   65          ;                UPON RECEIPT OF INTERRUPT, WRITE COMPLETE BIT CHECKED.
                   66          ;                IF SET, 'ENDWE' IS CALLED TO SHUT DOWN CONTROLLER.
                   67          ;                IF ILLEGAL ACCESS BIT SET, 'ILLACC' IS JUMPED TO.
                   68          ;                IF NEITHER BIT SET, 'BADINT' IS JUMPED TO INDICATING
                   69          ;                BAD INTERRUPT OCCURED.
                   70
                   71          ;       DATA PASSED:    NONE
                   72          ;       REGS AFFECTED:  NONE
                   73          ;       REQUIRES:       HOST PROGRAM MUST SET UP INTERRUPT VECTOR
                   74          ;                       SO 'HANDLE' EXECUTED UPON RECEIPT OF RST 6.5
                   75          ;                       COMMAND.
                   76          ;       CODE REQUIRED:  'ENDWE' SUBROUTINE CALLED TO SHUT DOWN
                   77          ;                       CONTROLLER AT END OF PROGRAM/ERASE CYCLE
                   78          ;       RAM USED:       1 BYTE - WRTCOM - WRITE COMPLETE STATUS BYTE
                   79          ;                                 - BIT 0 SET WHEN WRITE COMPLETE
                   80          ;
                   81          ;       I/O PORT USED:  PORT 21:
                   82          ;                         - BIT 0 = WRITE COMPLETE (ACTIVE HI)
                   83          ;                         - BIT 1 = ILLEGAL ACCESS (ACTIVE HI)
                   84
                   85
 000A              86 IOFMSK   EQU     1010B           ; MASK OUT INTERRUPT 6.5
 0021              87 WCPORT   EQU     21H             ; WRITE COMPLETE STATUS I/O PORT
                   88
                   89 HANDLE:
 0FE0 F5           90          PUSH    PSW             ; SAVE A-REG, FLAGS
 0FE1 DB21         91          IN      WCPORT          ; PICK UP CONTROLLER STATUS BITS
 0FE3 1F           92          RAR                     ; MOVE ILLEGAL ACCESS BIT INTO CARRY
 0FE4 DA1200    C  93          JC      ILLACC          ; GO TO ILLEGAL ACCESS ROUTINE IF BIT SET
 0FE7 1F           94          RAR                     ; MOVE WRITE COMPLETE BIT INTO CARRY
 0FE8 D21300    C  95          JNC     BADINT          ; IF NOT SET THEN GO TO BAD INTERRUPT HANDLER
 0FEB 3E0A         96          MVI     A,IOFMSK        ; UN-MASK 6.5 INTERRUPTS
 0FED 30           97          SIM
 0FEE CD0000    E  98          CALL    ENDWE           ; SHUT DOWN CONTROLLER
 0FF1 3E01         99          MVI     A,1H            ; SET WRITE COMPLETE INTERCOM BIT
 0FF3 320000    D 100          STA     WRTCOM          ; AND SAVE IN RAM                    AFN-01885A
```

Figure 29. Controller Interrupt Driver (Continued)

AFN-01913A

```
ISIS-II 8080/8085 MACRO ASSEMBLER, V3.0        MODULE   PAGE   3
  LOC  OBJ        LINE        SOURCE STATEMENT

  0FF6 F1          101         POP     PSW             ; RESTORE REGISTER
  0FF7 C9          102         RET                     ; AND RETURN BACK TO INTERRUPTED ROUTINE
                   103
                   104
                   105
                   106
                   107
                   108
                   109
                   110         CSEG
                   111
                   112
  0012 C7          113 ILLACC: RST     0               ; ILL ACCESS RESTART VECTOR FOR TESTING ONLY
  0013 C7          114 BADINT: RST     0               ; BAD INTERRUPT RESTART VECTOR FOR TESTING ONLY
                   115
                   116         END


PUBLIC SYMBOLS
HANDLE A 0FE0    WEDELY C 0000

EXTERNAL SYMBOLS
ENDWE  E 0000

USER SYMBOLS
BADINT C 0013    ENDWE  E 0000    HANDLE A 0FE0    ILLACC C 0012    IOFMSK A 000A    IONMSK A 000D    LOOP   C 0009
WCPORT A 0021    WEDELY C 0000    WRTCOM D 0000
ASSEMBLY COMPLETE,   NO ERRORS                                                              AFN-01885A
```

Figure 29. Controller Interrupt Driver (Continued)

AFN-01913A

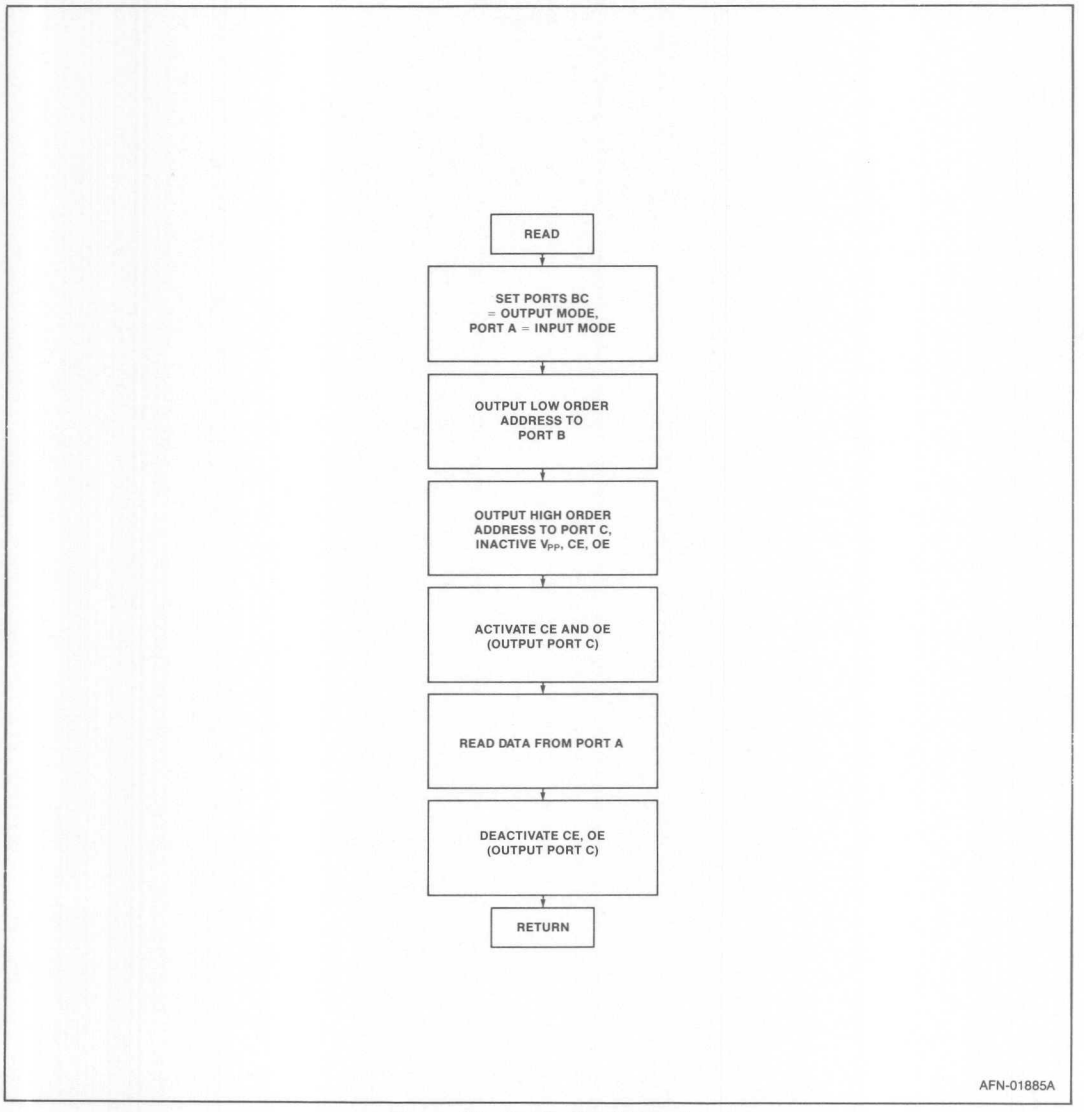


Figure 30. Controller IV Read

April 1981

Programming E²PROM With a Single 5-Volt Power Supply

Henry Fung, John F. Rizzo
Special Products Division
Applications Engineering

The Intel 2816 is a new generation of non-volatile memory in which writing and erasing can be accomplished on board by providing a 21 volt pulse. Figures 1 and 2 show the wave forms for byte erase (or write) and chip erase respectively. In order to generate the V_{PP} pulse, a power supply with output voltage of +24V is needed. In a system environment where this voltage is not available, a switching regulator can be used to convert +5V into +24V. This Application Note will discuss the design and implementation of such a regulator.

With the advent of LSI technology, the design of a dc-to-dc converter has been greatly simplified. Figure 3 shows the circuit diagram for a voltage converter using a TL497 switching voltage regulator. The converter presented here is very low cost and is excellent for use in systems where 5 volts is the only supply available.

In order to familiarize the reader with the operation of such a converter, the following discussion is appropriate. The circuit operates as follows: the frequency at which transistor Q1 is switching is determined by capacitor C1. The converter output voltage is fedback to an internal comparator that controls the on and off time of Q1. When Q1 is turned off, voltage across the inductor inverts, and the blocking diode CR1 is forward biased to provide a current path for the discharge of the inductor into the load and filter capacitors (C2 and C3). During the time when Q1 is turned on, the current into the inductor increases linearly. The blocking diode CR1 will become reverse biased and the output load current is provided by the filter capacitors. Figure 4 shows the waveform of the current into the inductor when the output is drawing 80mA. As can be seen, there is no gap between the charge and discharge cycles. Therefore, any current output exceeding 80mA will cause the output voltage to start losing regulation. The switching regulator efficiency can be calculated as a ratio of output power to input power. Therefore,

$$\% \text{ efficiency} = \frac{\text{Output power}}{\text{Input power}} \times 100\%$$

$$= \frac{24V \times 80mA}{5V \times 1160mA \times 0.5} \times 100\%$$

$$= 66\%$$

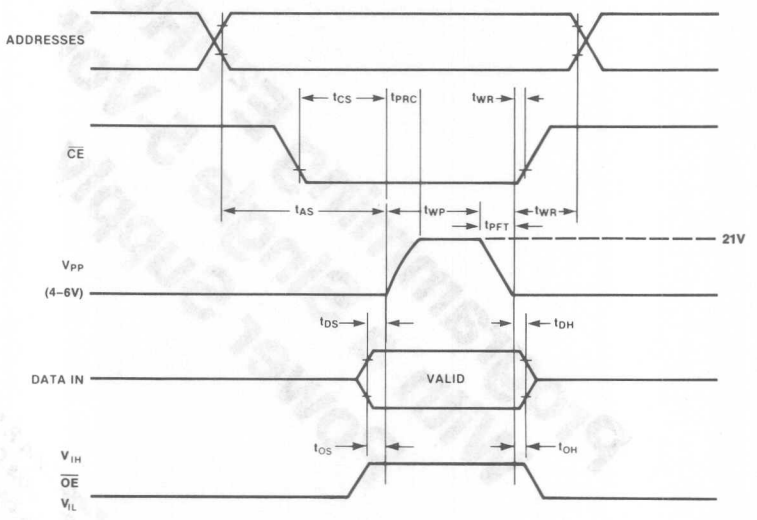


Figure 1. Byte Erase (or Write) Waveforms

The output voltage from the switching regulator can now be used to generate the V_{PP} pulse required to program the 2816 E^2PROM. Figure 5 shows the V_{PP} switch circuit diagram. CR1 is used to suppress any noise on the +24V. A2 is an open-collector gate. When its output is low, C1 and pin 5 of A1 will be shorted to ground. Therefore, Q1 will be turned off and V_{PP} pulse will stay at V_{CC} less one diode drop. When a write cycle is initiated, output of A2 will be high for 10mS. This would allow the capacitor to charge. The time constant is determined by R1 X C1 = 600μsec. As soon as the voltage across the capacitor is charged up to the zener voltage, the feedback amplifier will force this voltage to remain constant. The final output voltage is adjusted by R2. Q1 provides the additional current drive capability up to 75mA and CR2 across pin 5 and 6 of A1 will ensure a glitchless V_{PP} pulse.

The 2816 has an inhibit mode which allows the device to be deselected during programming. It also means that the V_{PP} switch has to supply the I_{PP} standby current for the unselected devices. Table 1 shows the maximum number of devices that can be supported by the switching regulator in an 8-bit and 16-bit system. Because of the inhibit mode device selection, only one switch is needed for many devices in system.

The dc-to-dc converter and V_{PP} circuit provide an overall solution for programming the 2816 E^2PROM using a single +5V supply. With its high current drive capability, the V_{PP} switch should satisfy over 95% of the design requirements. Therefore, it is recommended that the circuit be implemented whenever +24V is not available. This circuit has also been designed and tested to operate over the full temperature range, just like the 2816.

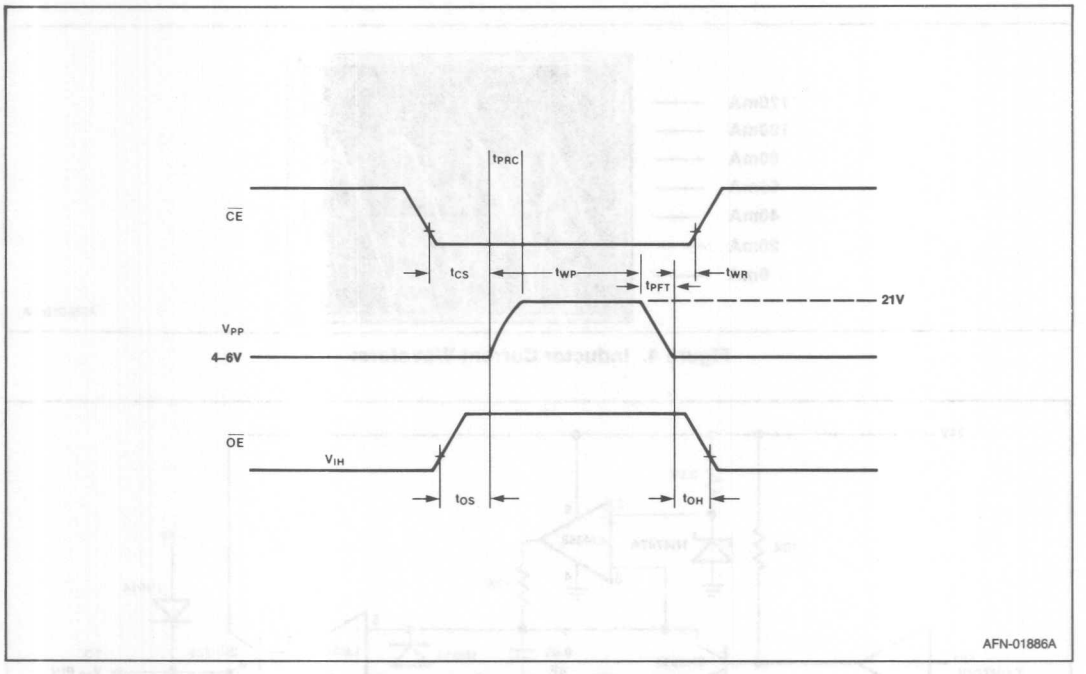


Figure 2. Chip Erase Waveforms

Table 1.

System	Active Programming Current	Standby Current	Devices Supported	K Bytes
8-bit	15mA	60mA	13	26
16-bit	30mA	45mA	10	20

NOTE: Total current (I_{PP}) = 75mA.

AFN-01913A

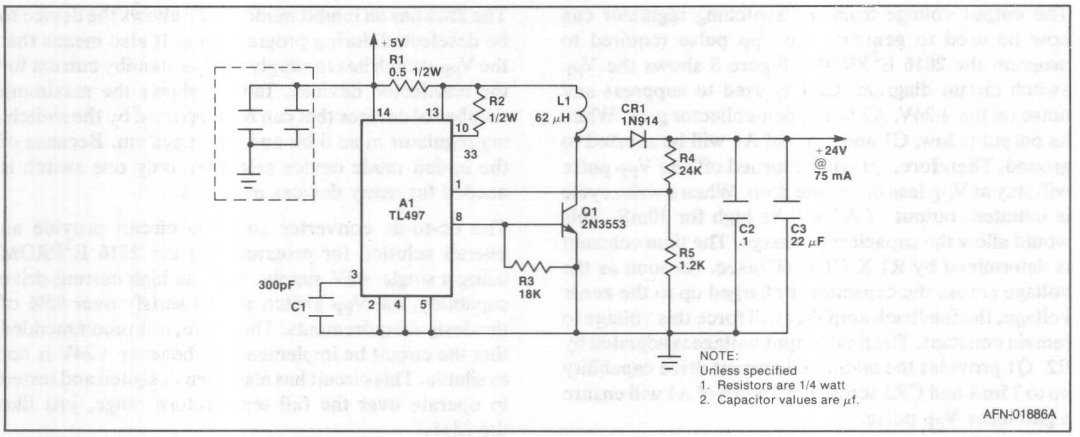


Figure 3. Step-Up Regulator Converts +5V into +24V

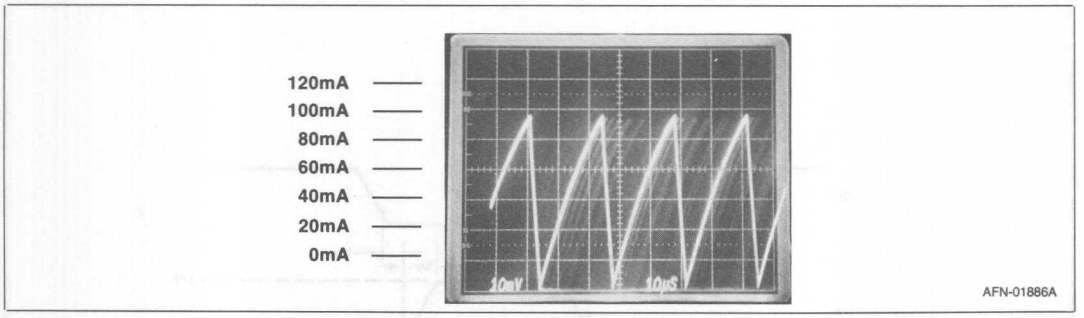


Figure 4. Inductor Current Waveform

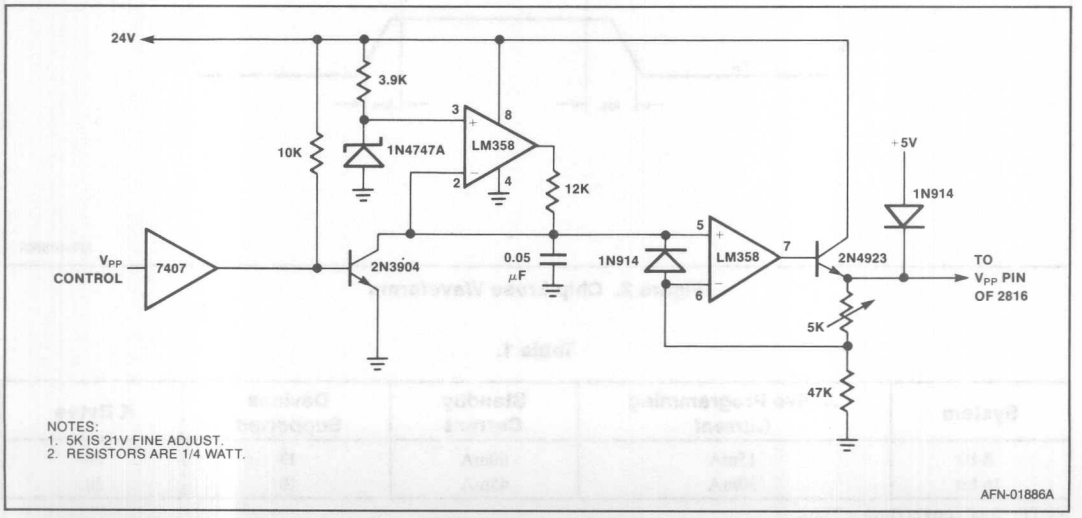


Figure 5. V_{PP} Switch

April 1981

Hardware and Software Download Techniques With 2816

John F. Rizzo and Randy Battat
Special Products Division
Applications Engineering

INTRODUCTION

Software Updates—how many times in microprocessor systems does software undergo revision? Unfortunately, many people say that it changes frequently. As we all know, such revision can be inconvenient, difficult and extremely costly. The 2816, E^2PROM, from Intel, can not only eliminate these expenses, but increase the functionality of your designs as well. The 2816 combines the benefits of ROM-like non-volatility with RAM-like flexibility. This application notes discusses the costliness of in-field software updates, how 2816 can solve these problems, and some circuit design information detailing how to implement an evolutionary system that eliminates current service costs.

IN FIELD SOFTWARE UPDATES

As technology progresses, the cost of microprocessor systems will become more dependent on design and service costs rather than component costs. Service costs today average about $100/hr. By 1985, assuming a typical inflation rate, those costs will approach $200/hr. Any necessary maintenance to change software, or adjust non-volatile parameters, adds hundreds of dollars to a typical system cost.

To take a realistic example, let's assume a typical microprocessor system (2000 in the field), with a service time of 2 hours per system. Also assume that each system needs to be updated a minimum of 2 times during the product's life. Given such assumptions, the cost involved is at least $400 per system. That's $800,000 for the total retrofit! If one assumes a doubling of labor rates in the next 5 years, the new retrofit cost would be $1.6 million. The 2816 can completely eliminate those costs.

By installing a remote software serial link, the software update can occur over telephone lines, free from service intervention. By 1985 service costs additional to each system will be as much as $800. Adding 2816 and a remote link to the system will cost about $50, a mere one-sixteenth the service cost. Today, a 40% savings can result. Figure 0 shows these cost trends.

It is clear that 2816 can save millions of dollars in maintenance costs. That is why it is such a cost effective solution to the many firmware update problems we face today.

In this application note, the hardware and software designs for such a solution will be discussed. First, though, let us examine the design criteria that are pertinent to the memory elements in such a system:

1) **NON-VOLATILITY**—data must be retained even when the host system is powered down.

2) **FAST ACCESS TIME**—With today's high speed microprocessor systems (i.e., the Intel 8086-2, the Zilog Z8000, and the Motorola MC68000) full throughput is only achieved with fast memory devices. For example, a high performance 8086-2 system for zero wait state operation requires a read access time of 250 ns.

3) **HIGH DENSITY**—As software costs rise, high-level languages will be used to reduce design time. Such high-level languages are often memory intensive, requiring high density memory chips to effectively contain dedicated system programs without sacrificing printed circuit board space.

4) **READ MOSTLY OPERATION**—Program memory and certain types of data memory are mostly accessed in a read mode. There are situations, however, where it is necessary to re-load an entire program (as in the case of a software revision), or reconfigure portions of data storage (e.g., when only certain parameters need to be changed). In these cases, the ability to write to the memory in-circuit is essential.

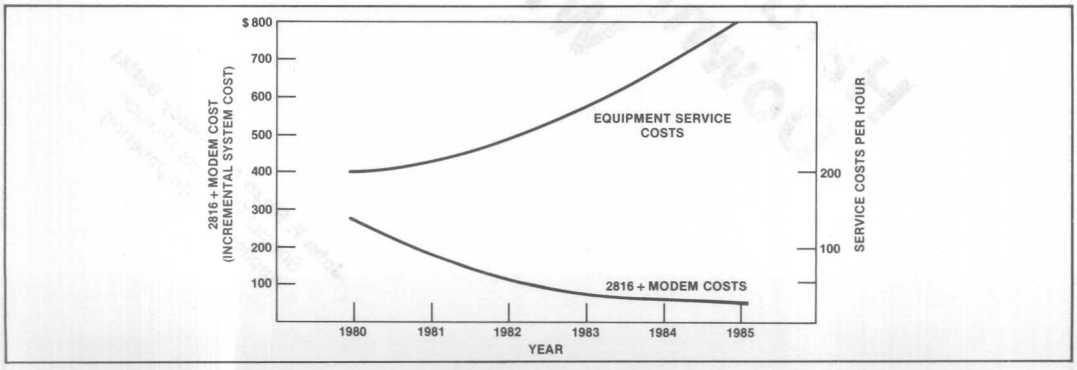


Figure 0. Service Cost Trends

The Intel 2816 fills the need for all these user requirements. It is truly non-volatile, offering greater than 20 year data retention. Access time is 250 ns, which is compatible with today's high speed microcomputer systems. The 2816 is electrically erasable on a per byte or per chip basis—a true read mostly memory, and it offers users 16,384 bits of storage organized as 2048 8-bit bytes.

Specific topics included in this Application Note are the philosophy behind downline loading, as well as the wide spectrum of application possibilities. Included here are four configuration examples. A discussion of both receiving and transmitting functions follows the examples.

DOWNLINE LOAD PHILOSOPHY

The E^2PROM is an excellent medium for storing non-volatile program and data information. The fact that it allows in-circuit erase and write suggests many possibilities as to the information source that the 2816 can be written from. In many instances, E^2PROM memories will be written from remote data facilities.

The telephone is an ideal means of transferring such information, since it is readily available and requires no special interface. With use of an acoustic coupler, serial binary data is converted into high and low frequency tones, which can be transmitted over a datacom link world-wide. Modems interface easily with microprocessors, and the software overhead of performing a downline load operation is minimal.

2816 REMOTE CONFIGURATION OPTIONS

Programs downline loaded to E^2PROMs find many applications in both large and small microcomputer systems. All configurations require a modem to interface electrical signals from a central processor with the acoustically driven telephone. Automatic modems are usually dedicated to a specific telephone line and are completely operated by a host processor. Manual modems are usually portable, relying on the human operator to physically place a telephone receiver in an acoustic coupler cradle, thereby closing the communication loop. Both automatic and manual modems can be used in E^2PROM-telephone communication systems, resulting in four possible configurations:

Manual Receiver — Manual Transmitter

This is a cost effective solution when telephone transmission is not performed often enough to warrant a dedicated telephone line and microprocessor system. Applications include infrequent field updates of program store, where a field system user would call a central factory to have 2816 memory devices reloaded.

Manual Receiver — Automatic Transmitter

Here an automatic transmitter is connected to a microprocessor system which answers the phone and transmits information to 2816s located in remote areas. Applications include field updates, as previously discussed, though a human operator on the transmitting end is not needed. This is advantageous when many field systems will be calling the central factory.

Automatic Receiver — Manual Transmitter

In this situation a microcomputer system would automatically answer the phone to receive information which will eventually be loaded in E^2 devices. This configuration could be used in remote, unattended systems, such as a microprocessor's controlling remote communications switches or repeaters. If parameters need to be changed, the remote switching processor would be telephoned and new parameters transmitted to the E^2PROMs in the system. This application exploits the byte erase feature of the 2816. Only those E^2 locations containing parameters to be changed need be rewritten.

Automatic Receiver — Automatic Transmitter

Fully automatic systems are useful when it is desirable to eliminate the need for a human operator. Here an auto-dial modem is used (previously discussed automatic systems use auto-answer modems). A central computer could be requested to call many remote units to automatically implement program or data update in E^2 memory without human intervention.

To provide an example of one of the four configurations described above, consider a manual receiver-automatic transmitter system. Because the hardware elements of an automatic transmitter are the same as those of an automatic receiver, by considering one example system, all four configurations can be described. With the example that will be discussed, the human operator is on the receiving end and initiates transmission by dialing the transmitter and placing a telephone receiver in an acoustic coupler cradle. The transmitter answers the telephone and transmits data to the receiver which eventually is loaded into E^2PROMs.

RECEIVER

A block diagram of the receiver system is shown in Figure 1. Three elements are of interest here: the modem and modem interface, the receiver CPU and associated software, and the 2816 and E^2 controller.

The receiver CPU is connected to a simple modem which converts serial binary data into acoustical tones. The standard Bell 103 modem or equivalent provides a host system with serial input/output data and various

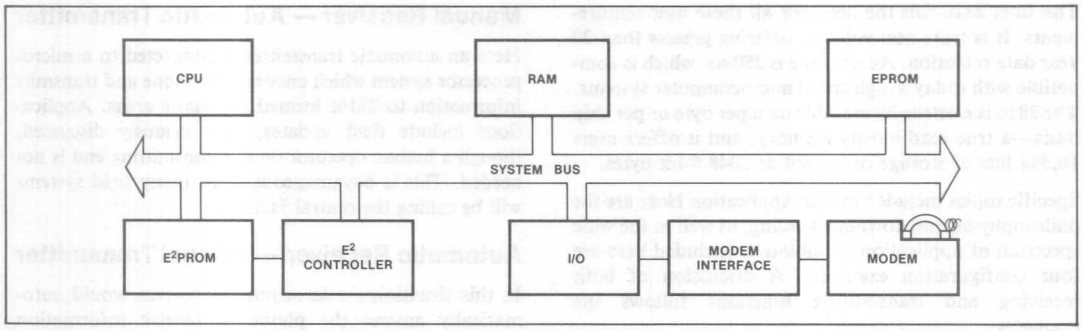
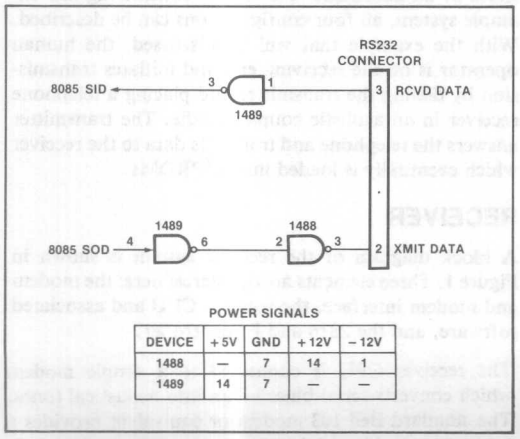


Figure 1. Typical MPU System With E²PROM Memory and Acoustic Coupler

status indicators (such as "carrier detect" which is active when a remote modem carrier signal is detected). The hardware required is minimal since a standard modem can be readily purchased. An RS232 interface is needed to interface 5V TTL signals from a CPU I/O port (or serial data line) to the ±12V RS232 compatible signals of the modem. The rest of the downline load operation is handled in software.

Figure 2 shows a simple modem interface. The MC1489 converts RS232 levels to TTL levels, while the MC1488 converts TTL signals to RS232. In the circuit shown, serial data I/O lines can be passed directly to a UART (Universal Asynchronous Receiver/Transmitter) for serial-parallel data conversion. Another option is to perform the serial-parallel conversion in software. If an 8085 processor is used, the serial I/O lines can be connected to the 8085 SOD and SID ports. The software required is also simple. The receiving CPU only needs to receive data bytes (possibly after a transmitter identification message is received) and program the E²PROM.



DEVICE	+5V	GND	+12V	−12V
1488	—	7	14	1
1489	14	7	—	—

Figure 2. A Simple Modem Interface

Figure 3 contains a flow chart outlining the process of receiving data. The processor first transmits an identifier message, then looks for a return identification message sent from the remote transmitter. This latter message may consist of a sequence of binary or ASCII data detailing the location of the transmitter, date and time of transmission, the number of bytes to be transmitted, the address in E²PROM of where data is to be located, etc. Next, the processor receives a data byte

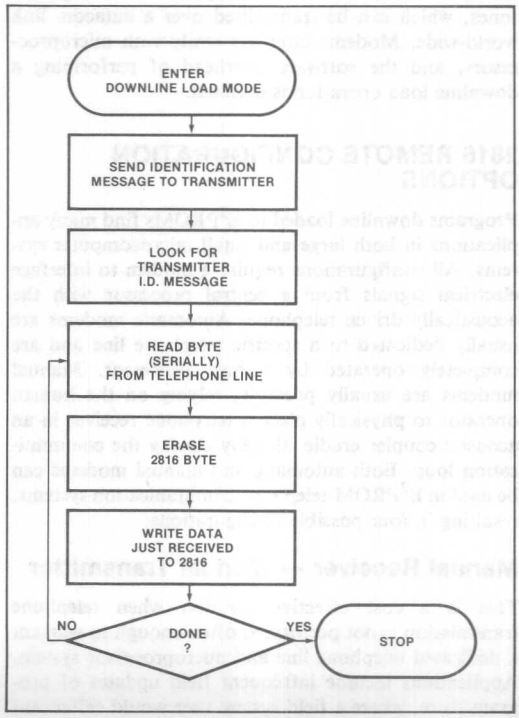


Figure 3. Receiver Software

which may be immediately programmed into the 2816 or saved temporarily in RAM. If serial-to-parallel data conversion is performed by software, data received must be saved in RAM. The 2816 cannot be programmed as each byte is received, since the processor must devote most of its time to receiving data bits and converting them to parallel form. However, if a UART circuit is used to perform data conversion in hardware, data bytes may be saved in E^2 memory as soon as they are received.

To illustrate this, assume data is transmitted at 300 baud (300 bits per second). Assuming each character consists of 1 start bit, 8 data bits, 1 parity bit, and 1 stop bit, then there are 11 bits per character so a character will be received every 36.7 msec. Between every character a 2816 byte must be erased (10 ms) and written (10 ms). Thus we spend 20 ms out of the 36.7 ms we have available during programming, while 16.7 ms of free time is left until the next byte is received.

The final consideration in the downline load receiver is a 2816 controller circuit. (AP102 describes several different controller configurations.) Controller I is convenient to use here. Figure 4 shows a block diagram of the circuit, while Figure 5 contains the circuit diagram. The read operation for the interface is identical to that for EPROMs. To read data, $\overline{CE}$ and $\overline{OE}$ are taken low after addresses are set up.

To write to the 2816, the host processor simply writes to memory. The controller circuit pulls the processor "ready" line low, stalling the CPU and stabilizing addresses and data for the 10 ms write interval while V_{PP} is active. The controller makes the 2816 resemble a slow write RAM except for the necessity of byte erase prior to writing.

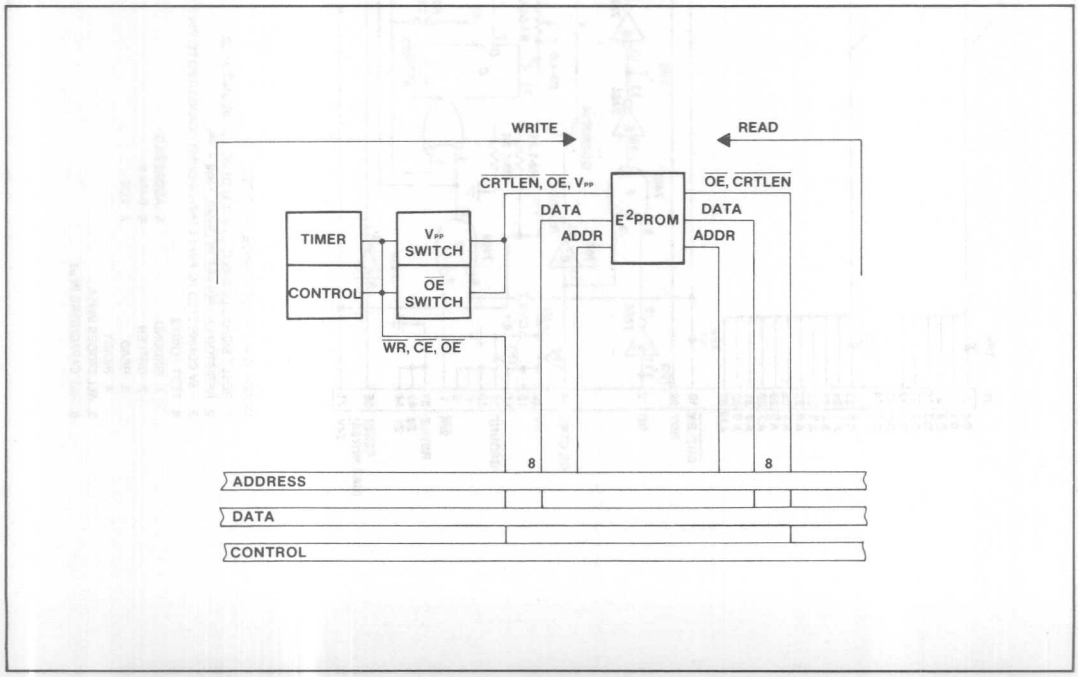
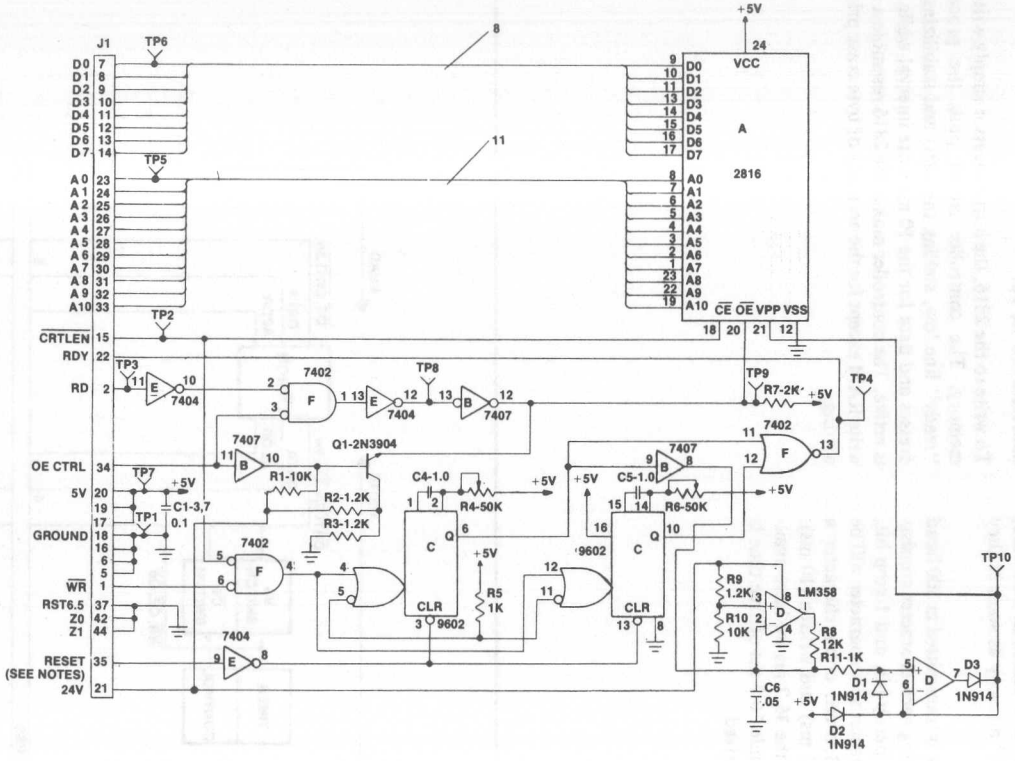


Figure 4. 2816 Controller Block Diagram

AFN-01913A



NOTES: (UNLESS OTHERWISE SPECIFIED)

1. RESET SIGNAL ORIGIN IS SYSTEM DEMONSTOR UNIT J1-22.
2. RESISTOR VALUES ARE IN OHMS, 1/4W, ± 5%.
3. +5V CONNECTED TO PIN 14 AND GROUND CONNECTED TO PIN 7 ON INTEGRATED CIRCUITS.
4. TEST POINTS
   1. GROUND
   2. CRTLEN
   3. READ
   4. READY
   5. ADDRESS 0
   6. DATA 0
   7. VCC
   8. OE CONTROL-READ
   9. RD
   10. VPP
5. ALL DIODES IN914.
6. ALL CAPACITORS IN μf.

Figure 5. E²DEMO Controller I

TRANSMITTER

The transmitter consists of a dedicated microcomputer connected to an auto-answer modem which in turn is attached to a telephone line. The transmit computer software, loops, waiting for an incoming call. When a call is received the modem is signaled to answer the telephone. Information, in the form of data bytes, is received and transmitted in the same fashion as is done on the receiving end. Essentially, all the base station must do is look for a remote processor identification message, send its own identification message, transmit data serially, and hang up the telephone. Additional features may also be implemented such as keeping a log of all calls received, their origins, etc.

Figure 6 contains a block diagram of a base station system. An 8085 processor is used, with an additional 512 bytes of RAM and 4K bytes of EPROM. A modem interface is shown, in addition to a keypad and display for local user operation, and a real-time clock for logging date and time information.

The EPROM memory contains program store and transmit information; i.e., the data that is to be transmitted to remote processor sites. Note that the transmit data EPROM could be replaced by an E^2 device to allow for frequent changes in transmission data without requiring the physical replacement of the transmit data store. RAM is used to save logging information, temporary program data, and a character input buffer which is used to store received characters when looking for a specific message.

The keypad/display module enables a local base station operator to interrogate the base station and reset date or time, access a call log, etc. The clock module is used to keep track of current date and time. Such data may be transmitted to remote processors, or may be used locally as a part of the information logged pertaining to each call received.

A modem interface is very similar to the receiver modem circuit shown in Figure 2. Figure 7 contains a circuit diagram of an auto-answer modem interface. The circuit provides all signals as that of Figure 2, but additionally converts the "Data Terminal Ready" signal and the "Ring Indicator" signal. "Data Terminal Ready" is provided by the host processor and tells the modem when to answer and hang-up the phone line. "Ring Indicator" is active when the phone line is ringing, and is used here to interrupt the processor.

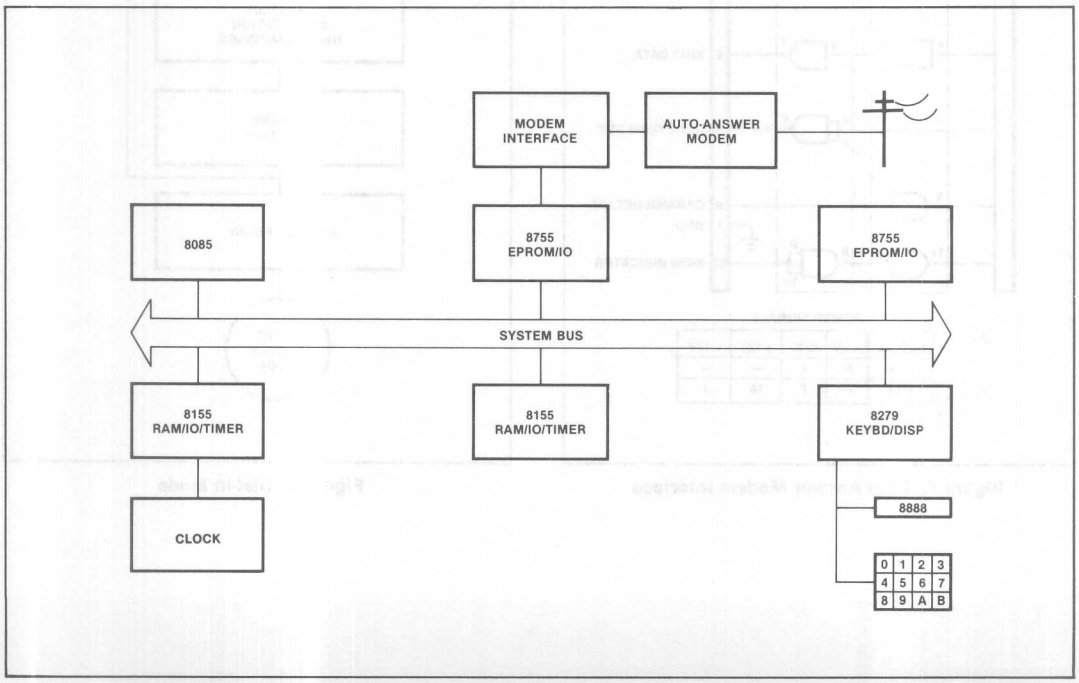
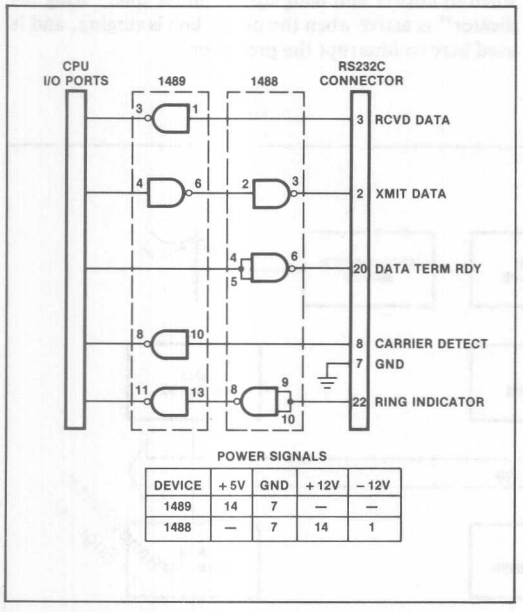


Figure 6. Base Station Block Diagram

Special Products Division Applications Engineering has constructed a base station similar to the one described here. It is used to transmit information to remote 2816s for demonstration purposes. In this unit, software consists of three operating modes:

- **Inactive Mode** is the default. The processor displays the time of day while waiting to enter one of the two modes described below.
- **Dial-In Mode** is entered whenever a call is received. A flow chart of Dial-In Mode software is shown in Figure 8. The processor answers the line, looks for a remote processor identification message, and transmits its own identification header, followed by a text data to be loaded in E^2PROM memory. The telephone is hung up as soon as transmission is completed, and inactive mode is entered.
- **Local User Mode** contains software to allow a local user to reset implemented via the local keypad/display.



POWER SIGNALS

DEVICE	+5V	GND	+12V	-12V
1489	14	7	—	—
1488	—	7	14	1

Figure 7. Auto Answer Modem Interface

CONCLUSION

Remote software changes—that's where 2816 is key. In this application note we've shown the costs involved in field software changes. The 2816 can eliminate field service and maintenance costs involved with software and constant changes. It can do this simply and cheaply through remote data links. Also discussed were typical circuit diagrams and system implementation. The bottom line is that 2816 can eliminate service costs in today's microprocessor systems.

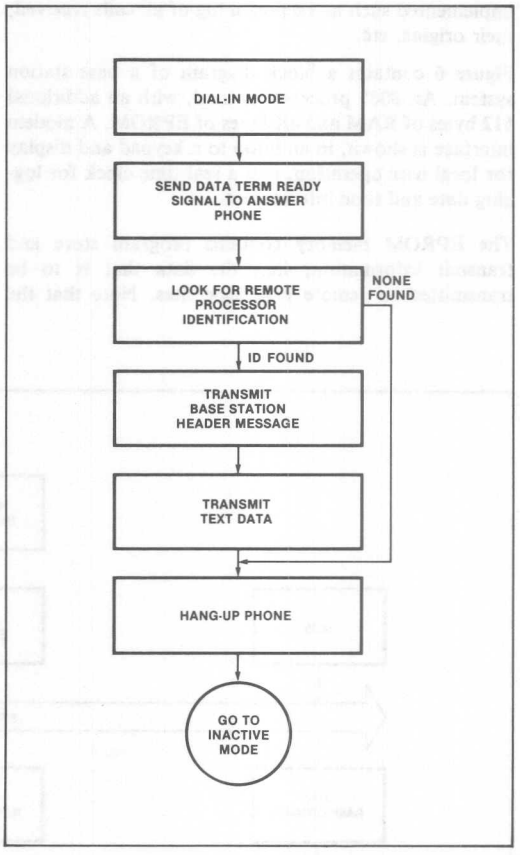


Figure 8. Dial-In Mode

August 1981

8298 Integrated E^2 Controller

Kevin Corbett, Larry Palley and Randy Battat
Special Products Division Applications Engineering

INTRODUCTION

The Intel 8298 is a special function peripheral device designed to interface Intel's full line of Electrically Erasable PROM memories to Intel microprocessor systems. It also enables the interface of E^2PROM devices to any system using an 8-bit I/O data bus. The 8298 is used to supervise the writing, erasing and reading of up to 8 2816 E^2PROMs. It is used because the write and erase cycles of E^2PROMs involve significant amounts of time (on the order of 10 ms each). A large portion of CPU time would be required to supervise these operations if no interface element existed. Adding the 8298 enables the supervision of these operations off-line, thereby freeing up the CPU and effectively increasing the throughput of the total system.

The 8298 is designed to enable a choice of hardware configurations. The use of the 8298 in the INDIRECT mode minimizes the hardware required to interface to the E^2PROMs, but sacrifices the ability to read and write the E^2PROMs directly. Adding extra hardware enables the 8298 to be used in the DIRECT mode, allowing read access at full system speed.

The goal of this application note is to facilitate simple and easy implementation of the 8298 and E^2PROMs by hardware and software engineers in their systems. Described herein are the 8298 and design configurations in which it can be used. The material is structured so that this application note can be a handy reference guide to the 8298 commands and to the two hardware configurations presented.

UPI-41A Base

The 8298 E^2 interface is based on the UPI-41A Universal Peripheral Interface. Special firmware has been written so that implementation of the E^2 interface can be undertaken in the simplest and most efficient manner. All timing signals for the 8298 are, therefore, identical to those of the UPI-41A. The use of the UPI-41A base also makes the 8298 fully compatible with all Intel microprocessors.

8298 HARDWARE

A pinout diagram of the 8298 and 8243 I/O expansion module are shown in Figure 1. The following section describes how these devices are hard-wired into the system. Each of the pin functions are described.

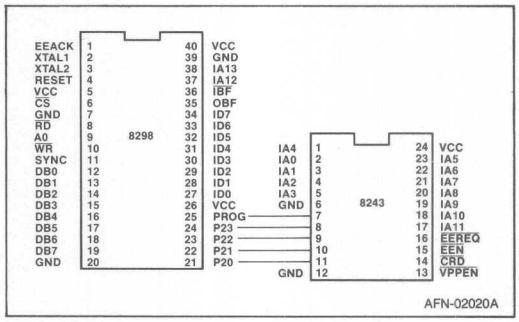


Figure 1. Pinouts of 8298 and 8243

System Interface

Interfacing the 8298 to the system is easily achieved. The 8298 is wired directly to the system bus as an I/O port with the data bus connected to the DBO-7 lines. All command, status, and data bytes are passed on this interface. The 8298 is addressed as a port using external I/O address decode hardware to enable the chip select ($\overline{CS}$) line. Only the A0 line is connected directly (or through a buffer) to the least significant I/O address line.

The A0 line selects between command and data input registers on writes to the 8298 and selects between the status and data output register during reads. By strobing the $\overline{RD}$ line of the 8298, data is enabled onto the data bus for reads. Writing data to the data bus occurs on the rising edge of the $\overline{WR}$ signal.

Two external pins are available for use as interrupts. The $\overline{IBF}$ line, when inactive, indicates that the 8298 is waiting for an additional command or data byte. The OBF line indicates to the processor that the data output register is full. The processor must service the 8298 during reads (when OBF active) by accessing data from the data output port. This allows the 8298 to continue reading the contents of the E^2PROMs. The $\overline{IBF}$, when activated during writes, indicates to the processor that it must halt data transfers to the 8298. The 8298 must write the contents of the data buffer to the E^2PROMS before the CPU continues to transfer data.

E^2PROM Interface

An array of 8 2816 E^2PROMS has 16K bytes of memory. The 8298 alone cannot supply the necessary 14 address lines for this array, so the 8243 I/O expansion module is used. The 8243 provides the twelve low-order address bits. The 8-bit internal data bus, and extra control pins are available from the 8298 directly. Adding the 8243 expander is implemented by connecting the

PROG and the P20-P23 lines of the two devices together. All necessary control and addressing data for the 8243 are passed from the 8298 through these lines.

Several control signals are necessary to read, write, and erase the E^2PROMS. These signals, supplied by the 8298 and 8243, are described individually below:

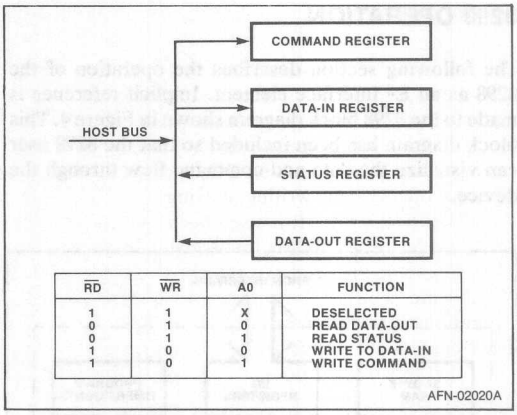
1. $\overline{EEN}$—E^2 enable is generated to access the E^2 array.
2. $\overline{CRD}$—Controller read is generated to indicate to the E^2PROM array that data is being read by the 8298.
3. $\overline{VPPEN}$—Programming pulse enable is generated to indicate that the programming pulse for a write should be applied to the E^2PROM array.
4. $\overline{EEREQ}$—E^2 request is generated to request access to the E^2PROM array in the direct configuration.
5. EEACK—E^2 acknowledge indicates that the E^2PROM array can be accessed by the 8298 in the direct configuration. It is generated by arbitration circuitry.

8298 Registers

The 8298 has four registers which the host CPU can access from the system data bus. Figure 2 shows the four registers and the method for selecting one of them. $\overline{RD}$, $\overline{WR}$, and A0 are all signals which are generated by the host CPU to perform register operations.

Command and Data Registers

The first byte of every command goes to the 8298 command register. The byte will be interpreted to determine the type of command and if additional bytes are required for the command. If a command has more than one byte, the additional bytes are sent to the data-in register. All data which is to be sent to the 8298 is written to the 8298 data-in register.



$\overline{RD}$	$\overline{WR}$	A0	FUNCTION
1	1	X	DESELECTED
0	1	0	READ DATA-OUT
0	1	1	READ STATUS
1	0	0	WRITE TO DATA-IN
1	0	1	WRITE COMMAND

Figure 2. 8298 Ports and Selection

When the 8298 performs a read operation, the data read by the host CPU is accessed through the data-out register.

Status Register

The 8298 status register contains a byte of information which describes the current state of the 8298. The host reads this register to determine if the 8298 is waiting for bytes of command or data, has received an illegal command, or is currently busy executing a command. Figure 3 and Table 1 describe the bit definition and bit interpretation of the status byte.

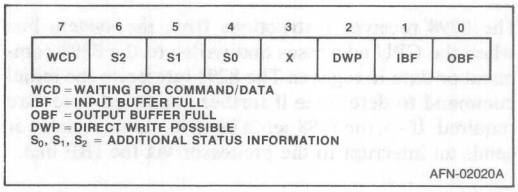


Figure 3. 8298 Status Word Bit Definition

Table 1. Status Definition

WCD	S2	S1	S0	OBF	DESCRIPTION
1	1	1	1	0	Waiting for Command
0	0	0	0	0	Executing Command
1	1	1	0	0	Illegal Command
0	0	X	X	1*	Illegal Command During Write
1	1	1	1	1	Read Command Complete
0	0	0	0	1	Read Data Ready
1	0	0	0	0	Waiting for Data Byte 1
1	0	0	1	0	Waiting for Data Byte 2
1	0	1	0	0	Waiting for Data Byte 3
1	0	1	1	0	Waiting for Data Byte 4
0	1	0	1	0	Write in Process

*Output buffer contains 10101010.

8298 OPERATION

The following section describes the operation of the 8298 as an E^2 interface element. Implicit reference is made to the 8298 block diagram shown in Figure 4. This block diagram has been included so that the 8298 user can visualize the data and command flow through the device.

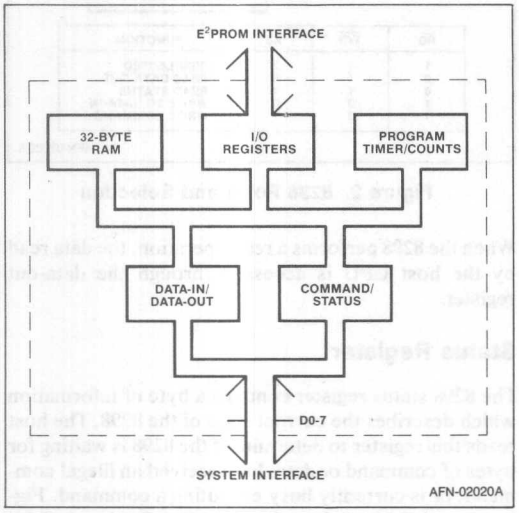


Figure 4. 8298 Block Diagram

The 8298 receives instructions from the system bus when the CPU addresses and writes to the 8298 command or data-in register. The 8298 interprets the initial command to determine if further command bytes are required. If so, the 8298 sets a bit in its status register or sends an interrupt to the processor via the $\overline{IBF}$ line.

After a full command string has been received, the 8298 determines whether the command is a read, write, erase or utility instruction. If it is a read or write command, data transfer begins. The CPU writes or reads data from the 8298 through the data input and output registers. This data is transferred directly to or from the E^2PROMS or buffered temporarily in 32 bytes of internal RAM, in the case of a multiple or series write. The 8298 indicates to the CPU when a data transfer is complete.

Series or multiple write transfers utilize the 32-byte internal data RAM buffer to speed system throughput. Using this buffer enables the CPU to write up to 32 bytes of data at system speed. The CPU uses this buffer by sending special write commands.

For commands where multiple bytes are written, the CPU sends a series of data bytes to the input register from which they are taken and stored in the buffer. When the buffer becomes full, the host processor can determine, via the status register, that data transfer to the 8298 must be temporarily suspended. The data in the full buffer is then written to the E^2PROMs, while the processor is freed for other processing. The 8298 notifies the CPU that data transfer can continue when the buffer is empty, with the IBF status or interrupt.

Read commands to the 8298 are all processed immediately. After receiving a read command, the 8298 accesses the E^2PROM array at the appropriate address and transfers that data byte directly to the data-out register. The host CPU is notified via the OBF status or interrupt to read this byte.

Utility commands are used to set up the 8298 configuration (DIRECT or INDIRECT), to initialize the timer associated with writing and erasing the E^2PROMs, to abort current commands, and to determine the address or data last written or read from the E^2PROMs.

CPU/8298 Communication

CPU communication with the 8298 can be implemented through software polling routines or through the use of interrupts.

In the polling routines, the CPU reads the contents of the 8298 status register at regular intervals to determine if data transmission is possible. When the status of the 8298 indicates that it is ready to accept a command or data byte or that it has data ready to be transmitted, the CPU can branch to a routine which performs these functions. The use of polling routine is easy to implement in software and requires no hardware overhead. Polling, however, takes a significant amount of CPU time. This time could be put to more efficient use.

The use of 8298 interrupts in place of polling routines eliminates wasted CPU time. The CPU can send a command string to the 8298 and let the 8298 execute it while it is devoting time to other tasks. When the 8298 completes command execution, it can interrupt the CPU. The CPU can then service the interrupt by returning to an 8298 communication routine.

A sequence in which a command string is transmitted to the 8298 is flowcharted in Figure 5. In this flowchart are two conditional states where WCD set and IBF clear are tested. A polling routine would be used to read the status register of the 8298, test these two bits, and loop

back if they are not set. An interrupt routine would be used to indicate the same state of these bits, but while the CPU is processing another task.

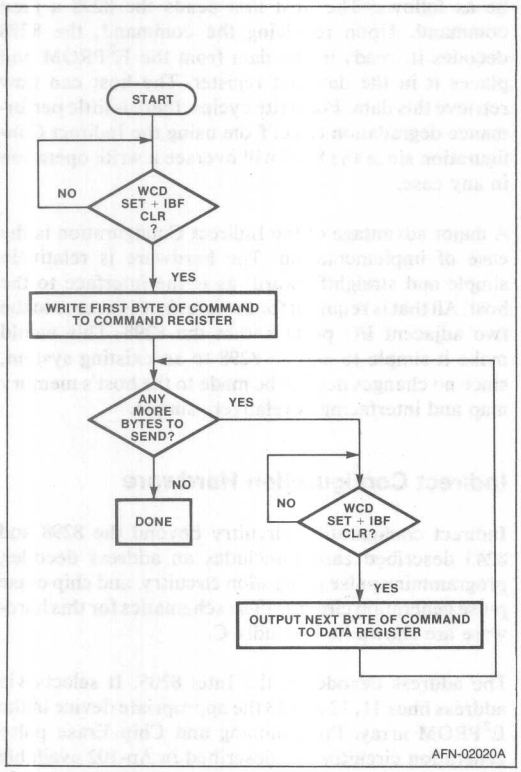


Figure 5. Flowchart for Command Transmission

Programming examples of this flowchart are shown in Figures 6 and 7. These routines are written in 8085 assembly language and are easily upgraded to 8086 code. The first set of routines is the actual module used to service the 8298. These routines are callable from a main program. The second set of routines are an example of the mulitple write command as it may exist in a main program. In this module the last statement is a call to the 8298 service routine in Figure 6 to write 10 data bytes to 10 different E^2PROM addresses.

SNDCMD and SNDDBX are subroutines used to send a command of one or more bytes to the 8298. They read the number of bytes required for the command from a command buffer (which the user has set up) and send them one at a time to the 8298. Each transmission monitors the status of the 8298 before taking place. CHKSTS is the polling used for this purpose.

This same module can be used in an interrupt-driven mode. The processor would initially call the SNDCMD routine to transmit the first command byte to the 8298. After returning to the other tasks, the CPU would later be interrupted by $\overline{IBF}$ and would send the next byte via the SNDDBX routine. Each ensuing interrupt issued by the 8298 would call the SNDDBX routine until all command bytes have been sent. The routine CHKSTS would not be needed when using interrupts. Significant savings in CPU time would result.

```
SNDCMD:  PUSH   PSW
         PUSH   B
         PUSH   D              ;SAVE REGISTERS

         LXI    H,CMDBUF       ;GET BYTE COUNT
         MOV    D,M
         INX    H
         MOV    E,M

         INX    H

         MVI    C,WCD
         CALL   CHKSTS

         MOV    A,M
         OUT    CMND           ;SEND COMMAND BYTE

         MOV    A,D
         ORA    A
         JNZ    DECD           ;TEST BYTE COUNT

         MOV    A,E
         ORA    A
         JZ     ENDSND         ;IF ZERO, DONE

DECD:    DCX    D
         MVI    C,DB1STS

SNDDBX:  INX    H              ;IF NOT ZERO, SEND

         CALL   CHKSTS         ;MORE DATA BYTES

         MOV    A,M
         OUT    DATA

         MOV    A,D
         ORA    A
         JNZ    DECD2

         MOV    A,E
         ORA    A
         JZ     ENDSND

DECD2:   DCX    D              ;CHANGE EXPECTED STATUS
         MOV    A,C            ;VALUE
         ADI    10H
         MOV    C,A
         JMP    SNDDBX

ENDSND:  POP    D
         POP    B
         POP    PSW            ;RESTORE REGISTERS
         RET

CHKSTS:  IN     STS            ;POLLING ROUTINE
         MOV    B,A
         IN     STS
         CMP    B
         JNZ    CHKSTS
         CMP    C
         JNZ    CHKSTS
         RET
```

AFN-02020A

Figure 6. 8298 Service Routine

```
; CONSTANTS
; PORTS
DATA    EQU   10H
STS     EQU   11H
CMND    EQU   11H
; STATUS BYTES
WCD     EQU   0F0H
DB1STS  EQU   80H
DB2STS  EQU   90H
DB3STS  EQU   0A0H
DB4STS  EQU   0B0H
; COMMANDS
SWCMND  EQU   25H
MWCMND  EQU   0CH
; BUFFERS
CMDBUF  DB    0F005H
; LONG ENOUGH FOR ANY COMMAND
START:  LXI   H, CMDBUF
        MVI   M,0          ;LOAD TOTAL NUMBER OF
        INX   H            ;BYTES TO BE SENT
        MVI   M,32
        INX   H
        MVI   M, MWCMND    ;LOAD COMMAND BYTE
        INX   H
        MVI   M,10         ;NUMBER OF DATA BYTES
        MVI   D,10H        ;ASSUME WE ARE
        MVI   E,0          ;WRITING TO EVERY 10TH
        MVI   C,0          ;ADDRESS STARTING AT 1000H
                           ;ASSUME WE ARE USING THESE
                           ;LOCATIONS FOR ERROR LOGGING
                           ;AND ARE NOW RESETTING THEM TO
                           ;ZEROS.
LDBFLP: INX   H
        MOV   M,D
        INX   H
        MOV   M,E
        INX   H
        MVI   M,0
        INR   C
        MOV   A,C          ;HAVE LOADED ALL
        CPI   10H          ;TEN BYTES, CAN NOW
        JZ    SNDERR       ;SEND COMMAND
        MOV   A,E
        ADI   10
        MOV   E,A
        JMP   LDBFLP
SNDERR: CALL  SNDCMD
        RET
```

AFN-02020A

Figure 7. 8298 Multiple Write Routine

SYSTEM CONFIGURATIONS

The two E^{2}PROM-8298 configurations presented in this section give the systems designer an option of minimal overhead hardware versus minimum E^{2}PROM access time. Each is described in detail including a discussion of the relative advantages and disadvantages to the user.

Indirect Configuration

The indirect configuration is so named because the host accesses the E^{2}PROMs indirectly—the 8298 acts as a buffer in all transmissions between the host and the E^{2}PROMs. In this hardware implementation, all write and read operations must be executed from the 8298, making full use of its input and output capabilities.

The effect of going through the 8298 is to slow the retrieval of data from the E^{2}PROMs. The host cannot read the data directly at system speed from the E^{2}PROM. A general example of a read operation would be as follows: The host first sends the 8298 a read command. Upon receiving the command, the 8298 decodes it, reads in the data from the E^{2}PROM and places it in the data-out register. The host can now retrieve this data. For write cycles, there is little performance degradation effect from using the Indirect Configuration since the 8298 will oversee a write operation in any case.

A major advantage of the Indirect Configuration is the ease of implementation. The hardware is relatively simple and straightforward, as is the interface to the host. All that is required for the interface is access to the two adjacent I/O ports within the 8298. This would make it simple to add an 8298 to an existing system, since no changes need to be made to the host's memory map and interfacing is relatively simple.

Indirect Configuration Hardware

Indirect configuration circuitry beyond the 8298 and 8243 described earlier includes an address decoder, programming pulse generation circuitry, and chip erase pulse generation circuitry. The schematics for this hardware are shown in Appendix C.

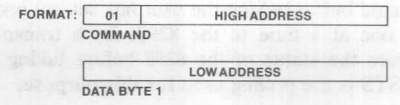
The address decoder is the Intel 8205. It selects via address lines 11, 12 and 13 the appropriate device in the E^{2}PROM array. Programming and Chip Erase pulse generation circuitry are described in Ap-102 available from your Intel representative.

Indirect Configuration Commands

Though the hardware implementation of the Indirect Configuration is relatively simple, there are numerous commands available that make the 8298 versatile to use. The following discussion covers all commands that may be used with this configuration.

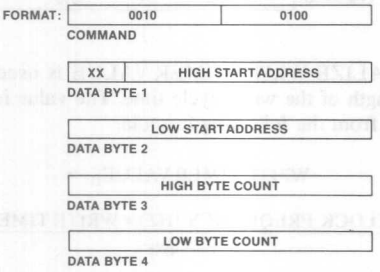
READ COMMANDS

There are two read commands available to send to the 8298. Both return the data to the data-out register of the 8298. INDIRECT READ is used to read a single byte of data from an E^{2}PROM. Note that the command and high address are combined in one byte.



AFN-02020A

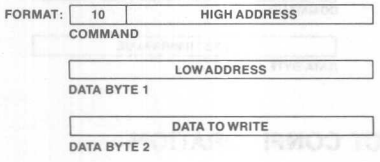
SERIES READ is used to read any number of sequential data bytes up to 16K bytes. The command consists of five bytes, so that it is usually more efficient to use SERIES READ for reading 3 or more bytes.



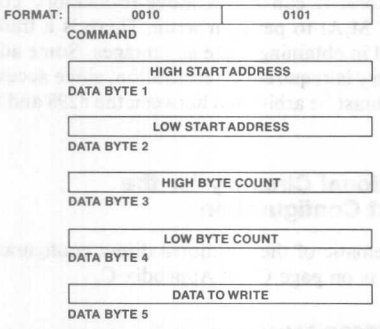
WRITE COMMANDS

There are three commands for writing to the E^2PROM via the 8298. They each have functions to make them useful for different applications.

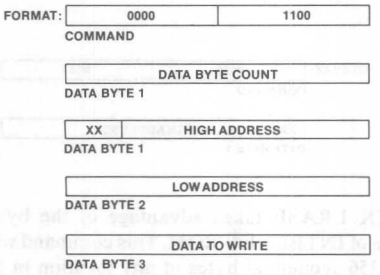
INDIRECT WRITE is similar in format to INDIRECT READ; it is for writing a single byte.



SERIES WRITE is the counterpart to SERIES READ. Note that once the command and data bytes have been sent, the 8298 asks for data byte 1 until "count" number of bytes have been sent. Up to 32 bytes of data are buffered in RAM by the 8298. Thus the data can be sent in a "burst."



MULTIPLE WRITE has no counterpart READ command. It is similar to INDIRECT WRITE in that it may write to nonsequential locations. The data and addresses are buffered in the 8298 so that up to 9 data/address groups may be sent in a burst fashion (i.e., no waiting for erase/write cycle).



The write commands have been optimized to increase throughput as much as possible. One example is the buffering of data (and possibly address) for SERIES WRITE and MULTIPLE WRITE. Another way is by reading first to check data at the address to be written . Under certain conditions, the byte erase can be avoided and thus 10 msec can be saved. If the data is identical with that to be written, the write can also be avoided. See Figure 8 below for the flowchart of the erase/write checking routine.

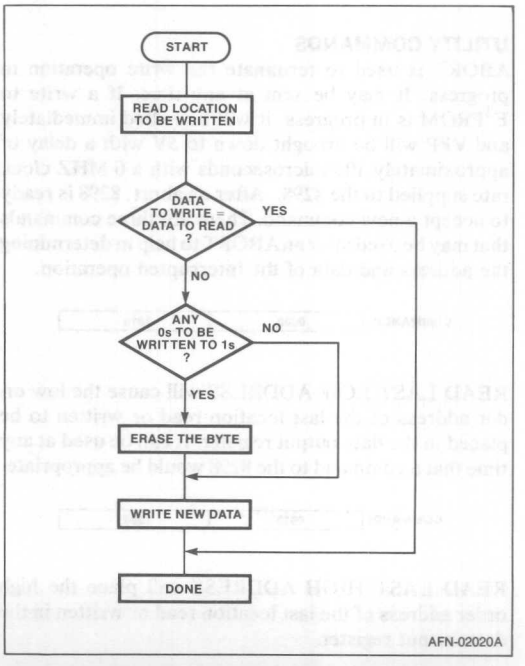
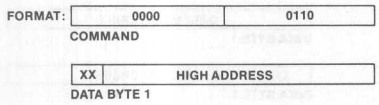


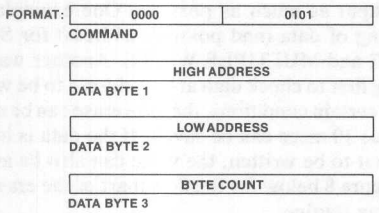
Figure 8. Erase/Write Flowchart

ERASE COMMANDS

There are two erase commands. CHIP ERASE will erase one complete E^2PROM. The chip to be erased is specified by sending the upper six address bits of the chip.

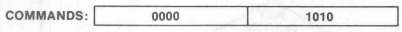


BLOCK ERASE takes advantage of the byte erase feature of INTEL's E^2PROM. This command will erase up to 256 sequential bytes at any location in the 16K array.

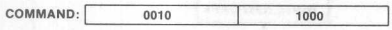


UTILITY COMMANDS

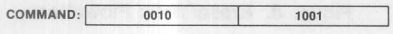
ABORT is used to terminate the write operation in progress. It may be sent at any time. If a write to E^2PROM is in progress, it will be halted immediately and VPP will be brought down to 5V with a delay of approximately 100 microseconds with a 6 MHZ clock rate supplied to the 8298. After an abort, 8298 is ready to accept a new command. There are three commands that may be used after an ABORT to help in determining the address and data of the interrupted operation.



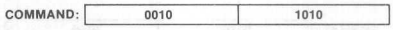
READ LAST LOW ADDRESS will cause the low order address of the last location read or written to be placed in the data output register. It can be used at any time that a command to the 8298 would be appropriate.



READ LAST HIGH ADDRESS will place the high order address of the last location read or written in the data output register.



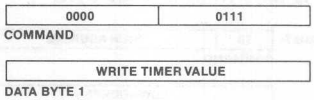
READ LAST WRITE DATA will put the data written to the E^2PROM during the last write cycle into the data output register.



INITIALIZE WRITE TIMER VALUE is used to set the length of the write cycle time. The value is determined from the following formula:

$$\text{WRITE TIMER VALUE}_{10} = 256 - \frac{(\text{CLOCK FREQUENCY (HZ)} \star \text{WRITE TIME (SEC)})}{480}$$

For example, with a 10 msec write time and a 6 MHz clock the timer value should be 131_{10}. Note: This is the default value (131_{10}). It is recommended that the 8298 be run at 6 MHz to achieve maximum efficiency. If it is run at a slower rate, the required value will have to be sent every time a reset is issued and on power-up.



DIRECT CONFIGURATION

Advantages/Disadvantages

The Direct Configuration allows the CPU to bypass the 8298 to read the E^2PROMS, thus taking advantage of the fast access times of the E^2PROMS. Write operations do not bypass the 8298, but the host can address the E^2PROM directly instead of through the I/O port. Therefore, it can use a move-to-memory command (MOV M,A) to perform write. There is a tradeoff involved in obtaining these advantages. Some additional circuitry is required for arbitration, since access to the array must be arbitrated between the 8298 and the host CPU.

Additional Circuitry for the Direct Configuration

A schematic of the additional direct configuration circuitry is on page C5 of Appendix C.

Host/8298 Interface

The major additions to the circuitry of the Indirect Configurations are the arbitration circuitry, and circuitry to force a write to the data-in register of the 8298

when a direct write is being made to E^2PROM address space. The 8298 recognizes this write as a direct write, gets the address from the additional address latches and completes the write. The arbitration circuitry consists of a D flip flop clocked by the SYNC output of the 8298. The SYNC output occurs once every time the 8298 performs an instruction. The 8298 sends out an EEREQ to see if it can access E^2PROMS. If it can, as determined by the arbitration circuitry, EEACK will sense a high level and the 8298 will continue the command. There are also latches for address retention and a transceiver for data in this configuration.

The addresses for direct reads or writes are strobed into these address latches which have outputs tied directly to the E^2PROM address pins. On a direct read or write the tristate outputs of these latches are enabled. Data can be read direct from the E^2PROM array using the data transceiver. The data being returned from a read operation goes through the transceiver when enabled by a read signal ($\overline{RD}$) from the host. All writes continue to be supervised by the 8298, so the host *must* read the status register of the 8298 before writing to the E^2PROMs to make sure that the 8298 is waiting for command.

Direct Configuration Commands

The added capabilities for the Direct Configuration are ease and speed of access for the host rather than a greatly extended command set. In fact, there are only two additional commands for the Direct Configuration.

ENABLE DIRECT WRITE is sent to inform the 8298 that the host may wish to use the "direct write." If this command isn't sent, and the host tries to do a direct write, it will be treated as an illegal command. If the host processor will be using direct writes, the ENABLE DIRECT WRITE command must be sent before attempting a direct write.

COMMAND:	0010	0010

DISABLE DIRECT WRITE signals the 8298 that the host no longer wishes to use the direct write utility.

COMMAND:	0010	0011

NOTE:
The 8298 defaults to this condition (on reset and power-up).

APPENDIX A
COMMAND SUMMARY

Table A summarizes the commands available to the host processor. It includes DIRECT READ and WRITE for completeness. The first column indicates the mnemonics for the command. The second column indicates the number of bytes in the command. The third column indicates the byte sequence, and the fourth shows the format of the byte. The last column shows the status which will be displayed in the 8298 status register when it is done processing the byte and ready to continue. The status mnemonics are defined in Table A.2.

Table A.1

Command	# of Bytes for Commands	Byte#	Format	Next Status
INDIRECT READ	2	1	01 High Address	DB1
		2	Low Address	OBF
SERIES READ	5	1	0010 0100	DB1
		2	XX High Start Address	DB2
		3	Low Start Address	DB3
		4	High Byte Count	DB4
		5	Low Byte Count	OBF
INDIRECT WRITE	2+ WRITE DATA	1	01 High Address	DB1
		2	Low Address	DB2
		3	DATA TO WRITE	WCD
SERIES WRITE	5+ WRITE DATA	1	0010 0101	DB1
		2	XX High Start Address	DB2
		3	Low Start	DB3
		4	High Byte Count	DB4
		5	Low Byte Count	DB1
			DATA TO WRITE	DB1/WCD NOTE 1
MULTIPLE WRITE	2+ ADDRESS/WRITE DATA	1	0000 1100	DB1
		2	Byte Count	DB1
		*	XX High Address	DB2
		*	Low Address	DB3
		*	DATA TO WRITE	DB1/WCD NOTE 2
BLOCK ERASE	4	1	0000 0101	DB1
		2	High Start Address	DB2
		3	Low Start Address	DB3
		4	Byte Count	WCD
CHIP ERASE	2	1	0000 0110	DB1
		2	XX High Address	WCD NOTE 3
ENABLE DIRECT WRITE	1	1	0010 0010	WCD
DISABLE DIRECT WRITE	1	1	0010 0011	WCD
ABORT	1	1	0000 1010	WCD
READ LAST LOW ADDRESS	1	1	0010 1000	WCD

Table A.1. Continued

Command	# of Bytes for Commands	Byte#	Format	Next Status
READ LAST HIGH ADDRESS	1	1	0010 1001	WCD
INITIALIZE WRITE TIMER VALUE	2	1 2	0000 0111 Write Timer Value	DB1 WCD

*ADDRESS + WRITE DATA

NOTES:
1. SERIES WRITE returns a status of DB1 whenever it is waiting for write data. When BYTE COUNT bytes have been sent, it will return a status of WCD. See SERIES WRITE command, under Indirect Configuration Commands, for more information.
2. MULTIPLE WRITE works in a loop format once the command and byte count are received. It requires groups of three bytes and cycles through the status as: -DB1—DB2—DB3- until "count" groups of bytes are received. It then returns a status of WCD.
3. High address is the high-order six bits of the highest address in an individual E^2PROM. An example might be: FOR 7FFH as the highest address in an E^2PROM, to erase would require sending 0000 0111 as the high address.

Table A.2. Status Bit Representations

Status Mnemonic	Binary Representation	Hex Representation
WCD	1111 0000	F0
DB1	1000 0000	80
DB2	1001 0000	90
DB3	1010 0000	A0
DB4	1011 0000	B0
OBF	0000 0001	01

NOTE:
If *DIRECT WRITES ARE ENABLED*, bit 2 will be a one (e.g., WCD would be 1111 0100B or F4H).

APPENDIX B
CONFIGURATION COMMAND SUMMARY

This Appendix contains a listing of the commands available to the 8298 E^2PROM controller in each of the configurations. It has been provided so that the user can easily look up available commands for their 8298 application.

Table B.1. Configuration and Applicable Commands

Commands	Indirect	Direct
Indirect Read	X	X
Series Read	X	X
Indirect Write	X	X
Series Write	X	X
Multiple Write	X	X
Chip Erase	X	X
Block Erase	X	X
Abort	X	X
Read Last Low Address	X	X
Read Last High Address	X	X
Read Last Write Data	X	X
Initialize Write Timer Value	X	X
Enable Direct Write		X
Disable Direct Write		X
Direct Write		X
Direct Read		X

APPENDIX C
HARDWARE SCHEMATICS

Mnemonic	Description	Origin/Schematic
A0–A13	System address bus	Host/C1, C5
D0–D7	System Data bus	Host/C1, C5
$\overline{\text{Reset}}$	System reset, active low	Host/C1, C3, C6
Reset	System reset, active high	— /C1
$\overline{\text{RD}}$	Host read, active low	Host/C1, C5
$\overline{\text{WR}}$	Host write, active low	Host/C1, C6
IO/$\overline{\text{M}}$	Host signal	Host/C1, C6
IA0–IA13	Internal address bus	8298 & 8243/C2, C5
ID0–ID7	Internal data bus	8298/C2, C5
EEN	E^2PROM enable, active high	8243/C3, C5, C6
$\overline{\text{EEN}}$	E^2PROM enable, active low	— /C2, C6
EEREQ	E^2PROM request	8243/C6
EEACK	E^2PROM acknowledge	8298(I)/C6
$\overline{\text{VPPEN}}$	V_{PP} circuitry enable, active low	8243/C4
$\overline{\text{CRD}}$	8298 read signal, active low	8243/C3
ALEQ	ALE (Qualified)	C6
HOSTACK	Host Acknowledge	C6/C5
$\overline{\text{HOSTRD}}$	Host Read of 2816s	C5
$\overline{\text{IBF}}$	Input Buffer (8298) not full, active low	8298/C1/C6
OBF	Output Buffer Pull	C1/C6
CS1, CS2	Chip select to enable 8298	C6
CERASE(ID7)	Chip erase signal	C3

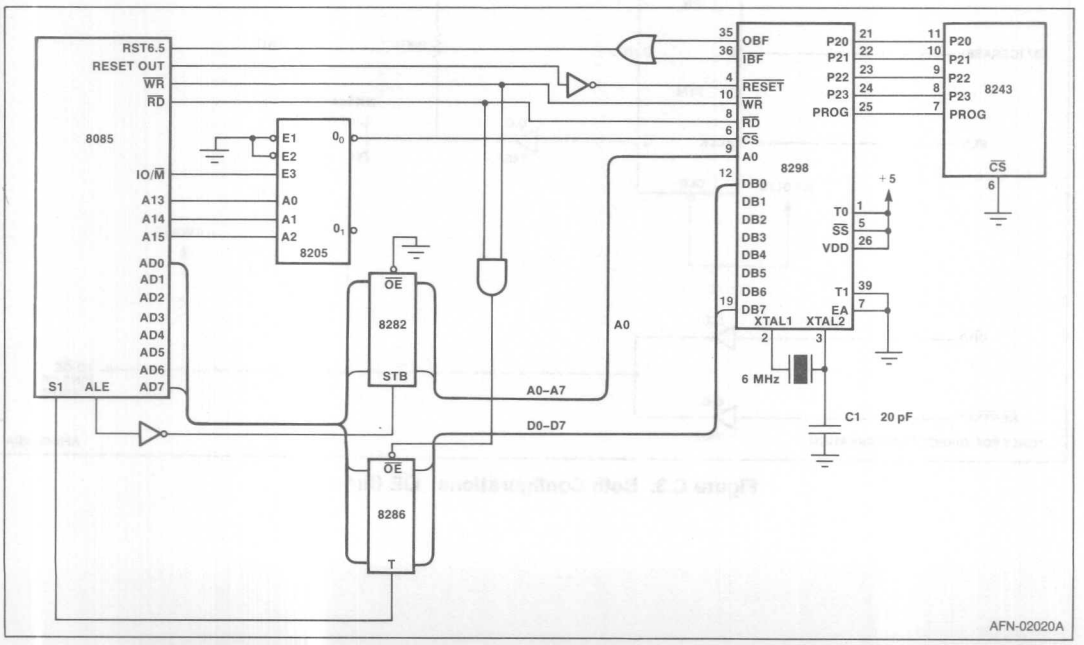


Figure C.1. Both Configurations: Host CPU/8298 Interface Example Using 8085

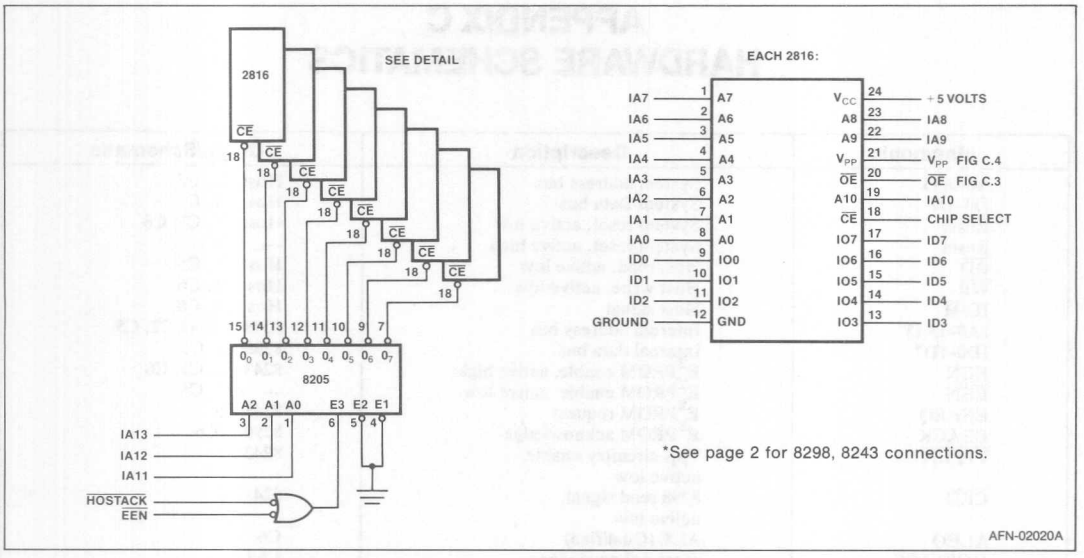


Figure C.2. Configurations: E^2 Array—Eight (8) 2816

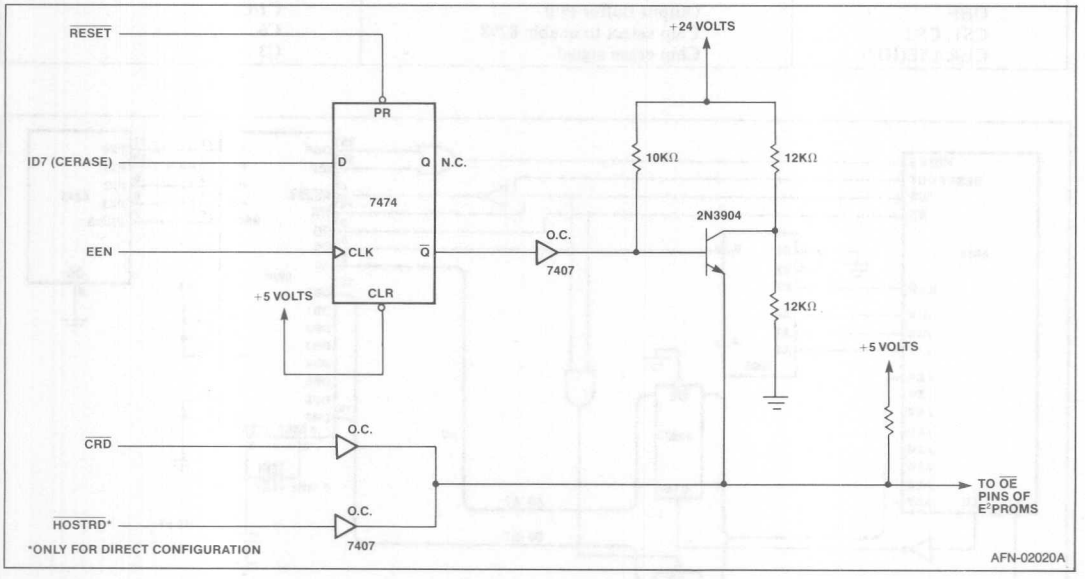


Figure C.3. Both Configurations: $\overline{OE}$ Circuitry

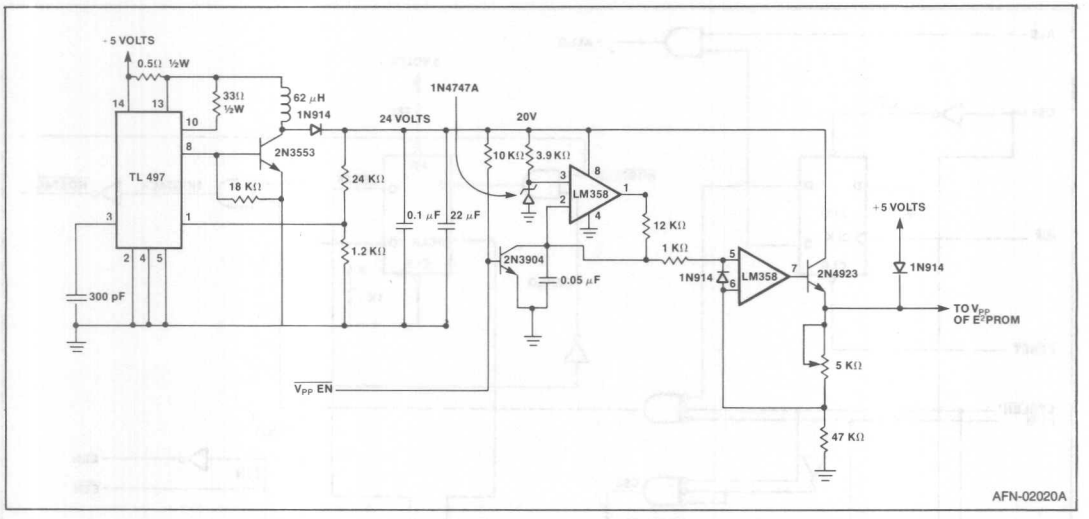


Figure C.4. Both Configurations: V_{PP} Circuitry (5 Volt Only—DC to DC Converter)

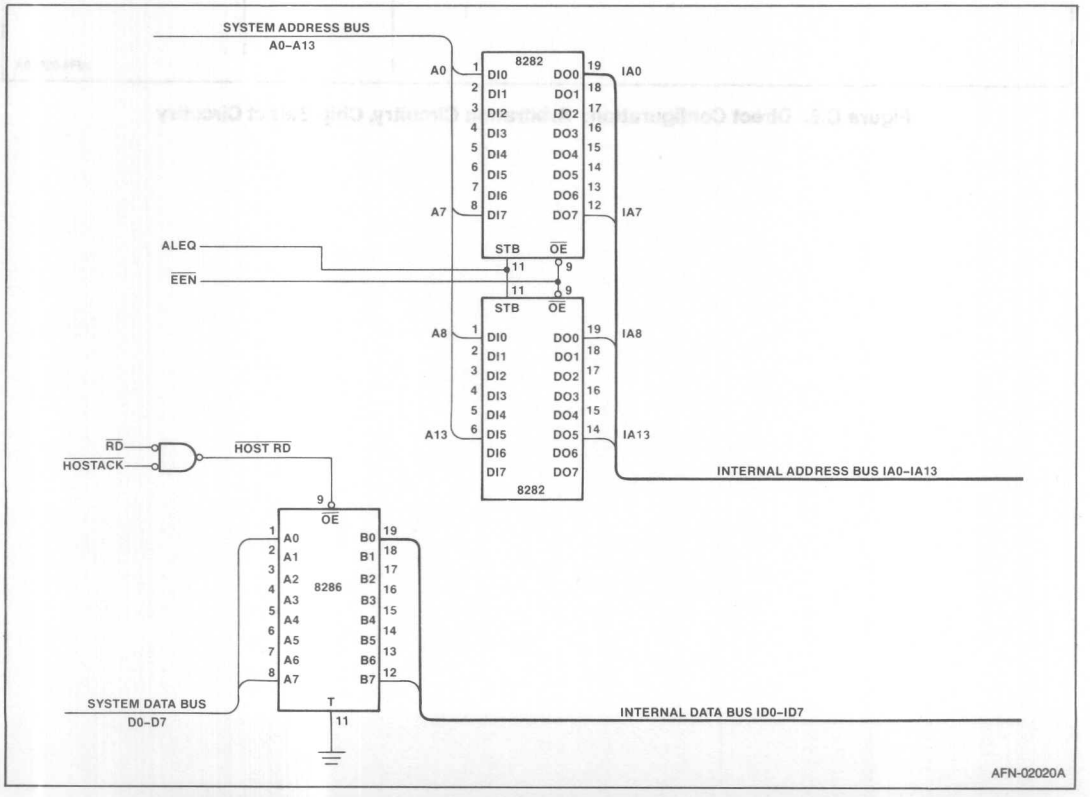


Figure C.5. Direct Configuration: Data/Address—Latches/Drivers

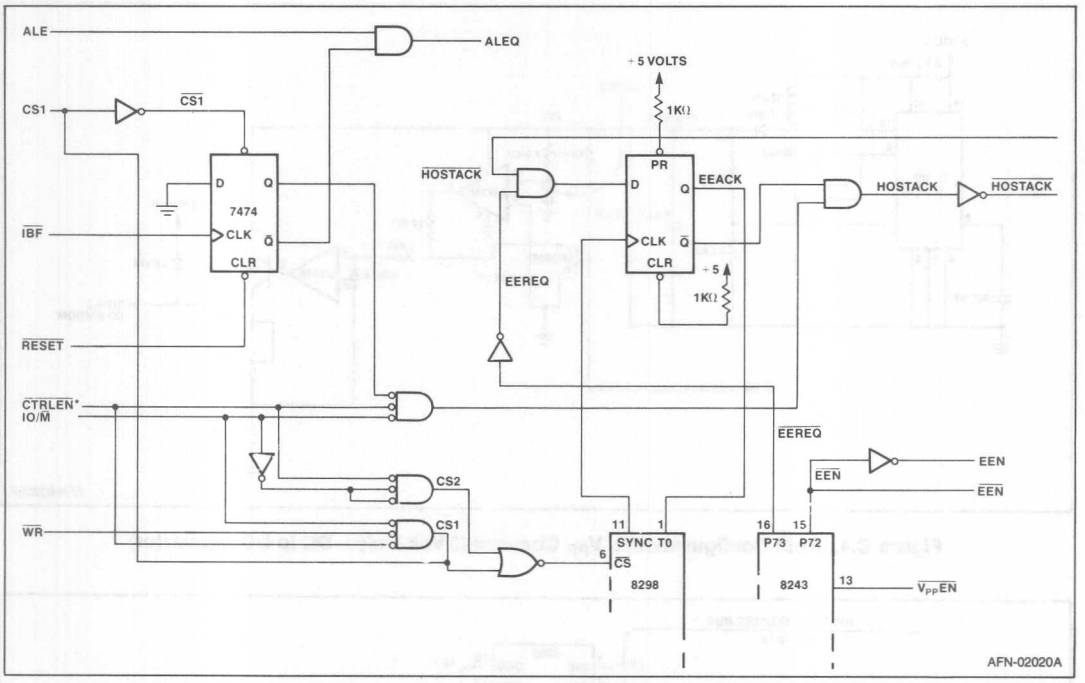


Figure C.6. Direct Configuration: Arbitration Circuitry, Chip Select Circuitry

intel

APPLICATION NOTE

AP-136

September 1981

A Multibus-Compatible 2816 E^2 PROM Memory Board

Justin Orion
Special Products Division
Applications Engineering

© Intel Corporation, 1981.

AFN-02067A

INTRODUCTION

The Intel Special Products Division E^2 Multibus Memory Board is an excellent example of how to implement the 2816 Electrically Erasable PROM in a multibus system. The board is completely Multibus compatible and can be plugged into any existing Intellec Microcomputer Development System. It can also be used in a Multibus-compatible system with any combination of Intel Single-Board Computer (iSBC) Modules. The memory board has a capacity of up to 16K bytes of electrically erasable non-volatile memory storage (8 2816s). The board can be read at microprocessor system speeds (250 ns). Writing to the E^2 Board requires only a single system write cycle. When the write operation is complete, the E^2 Board notifies the CPU by lowering an Interrupt Line. Individual 2816s can be erased in a similar manner with one write operation.

The E^2 Memory Board can operate with either an 8 or a 16-bit-wide data bus. This is determined by one jumper and by a 3628A Bipolar PROM used for decoding the addresses to the MOS PROM Array. The 3628A gives the board the capability of accommodating combinations of 2816s, 2815s, 2716s, 2732s, 2732As, and 2764s. The JEDEC pin compatibility of Intel's MOS EPROMs allows these devices to be plugged into the same 28-pin sockets. The 3628A, used as a decoder, allows these different memory size MOS PROMs to be used in the same array in various combinations. This enables the user to mix the devices in the memory array to fit the system's particular applications. The Multibus card can be located anywhere within up to one Megabyte of addressing space. The V_{PP} supply voltage is generated on board, and the only voltages needed are the standard Multibus +5V, +12V, and −12V power supplies. Finally, the E^2 memory system can be powered up and down repeatedly without losing one byte of its 16K bytes of data.

INSTALLATION INSTRUCTIONS

Below is a procedure to prepare the E^2 memory card for use. Following the list are detailed instructions for each step.

Procedure

1. Install the correct shorting plugs on the following jumper groups:

   J9–J12
   Board Address Location

   J13–J19
   RESET and Chip Erase I/O Addresses

   J1–J8
   Interrupt Line Selection

   J20
   Data Bus Width

   J21–J24
   PROM Socket Address Configuration

   JW1–JW8
   PROM/RAM Selection

2. V_{PP} Pulse Width Selection
3. Set switches SW1–SW5 according to the MOS PROM density used.
4. Adjust the V_{PP} high voltage level.
5. Select the proper $\overline{XACK}$ delay based on the t_{ACC} of the slowest MOS PROM used.

Board Address Location

The E^2 board can be assigned to any one of the 16 64K byte pages within a one-megabyte address space. If only 64K of address space is available, leave the jumper pairs J9–J12 open. Otherwise, install the shorting plugs to select the desired page as shown in the following chart:

Board Address Selection

X = install shorting plug
O = open

As shown in the assembly drawing in Appendix D, 2 pins reside at each jumper location. A shorting plug is simply inserted at the jumper location for installation.

64K Page	Jumper			
	J12	J11	J10	J9
0K– 64K	O	O	O	O
64K– 128K	O	O	O	X
128K– 192K	O	O	X	O
192K– 256K	O	O	X	X
256K– 320K	O	X	O	O
320K– 384K	O	X	O	X
384K– 448K	O	X	X	O
448K– 512K	O	X	X	X
512K– 576K	X	O	O	O
576K– 640K	X	O	O	X
640K– 704K	X	O	X	O
704K– 768K	X	O	X	X
768K– 832K	X	X	O	O
832K– 896K	X	X	O	X
896K– 960K	X	X	X	O
960K–1024K	X	X	X	X

RESET and Chip Erase Functions

The board requires two consecutive I/O addresses to control the RESET and Chip Erase functions. Doing an

I/O write cycle to one or the other address activates the particular function. The even I/O address selects the RESET function, an odd I/O address sets the Chip Erase function.

Jumpers J13 through J19 determine which two consecutive I/O addresses are to be used. The following chart shows the jumper scheme for I/O addressing.

	Jumper	J19	J18	J17	J16	J15	J14	J13	
	I/O Address bit	A7	A6	A5	A4	A3	A2	A1	A0
Function	RESET	b	b	b	b	b	b	b	0
	Chip Erase Mode	b	b	b	b	b	b	b	1

b = address bit

Once two consecutive I/O addresses for these two functions are determined, then install shorting plugs on J13–J19 corresponding to the 1's in the upper 7 bits of the two I/O addresses.

Examples:

for 00H = RESET
01H = Chip Erase,
install *no* shorting plugs

for 8AH = RESET
8BH = Chip Erase,
put shorting plugs on J13, J15, and J19

for 32H = RESET
33H = Chip Erase
put shorting plugs on J13, J16, J17

Interrupt Line Selection

The E^2 Memory Board generates an interrupt upon completion of a Byte Erase/Write or a Chip Erase operation. Any one of the 8 Multibus Interrupt Lines can be used. To select a given Interrupt Line, install the shorting plug indicated below:

Interrupt Line	Jumper
$\overline{\text{INT 0}}$	J1
$\overline{\text{INT 1}}$	J2
$\overline{\text{INT 2}}$	J3
$\overline{\text{INT 3}}$	J4
$\overline{\text{INT 4}}$	J5
$\overline{\text{INT 5}}$	J6
$\overline{\text{INT 6}}$	J7
$\overline{\text{INT 7}}$	J8

Data Bus Width

The Multibus Card can use either an 8-bit or a 16-bit-wide data bus. Jumper J20 should be installed for 16-bit data buses, and left open for 8-bit-wide buses.

In addition, switches SW1–SW5 must be set so that the correct MOS PROM(s) are enabled for upper and/or lower byte operations. Refer to the 16-Bit Data Bus Structure section for an explanation of the E^2 Board internal data bus structure. Refer to the following PROM Array Decoder subsection for directions on setting switches SW1–SW5.

PROM/RAM Selection

If E^2PROMs or EPROMs are used in the MOS PROM Array, do the following:

Install shorting plugs on the following jumpers:

JW1, JW3, and JW8

Leave these jumpers OPEN:

JW2, JW4, JW5, JW6, and JW7

V_{PP} PULSE WIDTH SELECTION

Ensure that the correct RC timing components are installed for the E^2PROM to be used:

E^2PROM	R3	C8	t_{WP}
2816	10 KΩ	4.7 μF	10 ms
2815	24 KΩ	10 μF	50 ms

R3 and C8 are located at the top and center of the E^2 board on the left of I.C. H1.

MOS PROM Array Decoder

(See Figure 1) The Bipolar PROM is used to select the MOS PROM or MOS PROM pair being addressed. The Bipolar PROM holds decoding algorithms for 2816s, 2815s, 2716s, 2732s, 2732As, and 2764s. The decoder also selects one or two devices at a time depending on whether an 8-bit or a 16-bit data bus is being used. The Dipswitch at the top of the board determines which decoding algorithm is to be used. Use Table 1 to choose the proper switch setting for the MOS PROMs to be loaded in the Array. The correct shorting plugs must also be installed on Jumpers J21–J24—see Table 2.

If it is desired to have devices of different densities in the Array or if a decoding algorithm other than the one provided is needed, the spare blocks in the 3628A can

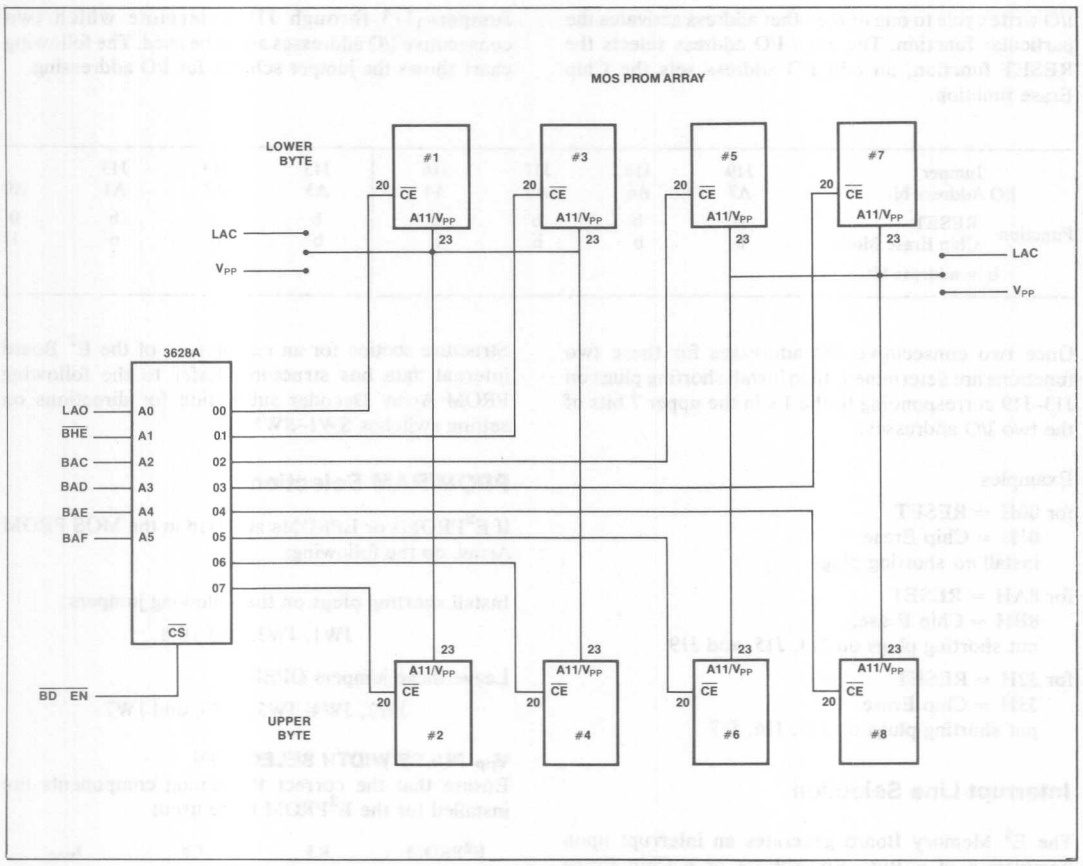


Figure 1. PROM Array Address Configuration

be programmed with new algorithms. The structure of the algorithm is determined by a simple principle. The output corresponding to the $\overline{CE}$ of the MOS PROM to be selected should be a logic 0 for the address range of that PROM. The selecting addresses are A12, A13, A14, A15 (called BAC–BAF on the schematic), and A0 and $\overline{BHE}$. (See section on 16-Bit Data Bus Structure for information on the use of A0 and $\overline{BHE}$.) The smallest address range is 2K bytes. Addresses A12–A15 select a pair of PROMs while A0/$\overline{BHE}$ select one or both of the two PROMs in that pair. Figures 2 through 7 can be used as examples. (Also see Appendix B.)

The decoding algorithms must also take into account the data bus width. See figures 2 through 4 for examples of 8-bit data bus algorithms and figures 5 through 7 for examples of 16-bit data bus algorithms. Note that the proper shorting plugs must be installed on Jumpers J21–J24 according to the device densities used. Jumpers J21–J24 simply connect address A11 or V_{PP} to pin 23 of the 28-pin MOS PROM socket. If a given Array half (sockets 1–4, for example) is to be loaded with 2816s or 2716s, then V_{PP} must go to pin 23 (Jumpers J22 and J24). If 4K or 8K byte parts are used, then All must be connected to pin 23 (Jumpers J21 and J22).

A few rules must be followed in mixed-density Arrays. (1) Each socket pair (1 and 2, 3 and 4, etc.: see Figure 1) must contain devices of the same density. (2) Each Array half (sockets 1–4 and sockets 5–8) can contain either 4K and 8K pairs, or 2K pairs, but not both.

If desired, a 3636B can be used instead of a 3628A. The 2K X 8 3636B will allow the encoding of twice as many decoding algorithms as the 1K X 8 3628A.

The blank PROM Decoder charts in Appendix G may be helpful in planning new decoding algorithms.

SYSTEM ADDRESS A0-15 HEX	DECODER CIRCUIT INPUTS BAF	BAE	BAD	BAC	$\overline{BHEN}$	LA0	$\overline{CE}$'S 7	6	5	4	3	2	1	0	BYTE
0 X X X	0	0	0	0	1	0							0	0	L
					1	1									H
1 X X X	0	0	0	1	1	0					0	0			L
					1	1									H
2 X X X	0	0	1	0	1	0			0	0					L
					1	1									H
3 X X X	0	0	1	1	1	0	0	0							L
					1	1									H
4 X X X	0	1	0	0	1	0									L
					1	1									H
5 X X X	0	1	0	1	1	0									L
					1	1									H
6 X X X	0	1	1	0	1	0									L
					1	1									H
7 X X X	0	1	1	1	1	0									L
					1	1									H
8 X X X	1	0	0	0	1	0									L
					1	1									H
9 X X X	1	0	0	1	1	0									L
					1	1									H
A X X X	1	0	1	0	1	0									L
					1	1									H
B X X X	1	0	1	1	1	0									L
					1	1									H
C X X X	1	1	0	0	1	0									L
					1	1									H
D X X X	1	1	0	1	1	0									L
					1	1									H
E X X X	1	1	1	0	1	0									L
					1	1									H
F X X X	1	1	1	1	1	0									L
					1	1									H
X=HEX DIGITS	A5	A4	A3	A2	A1	A0	0 7	0 6	0 5	0 4	0 3	0 2	0 1	0 0	
	ADDRESS INPUTS 3628A						OUTPUTS								

L = LOW BYTE
H = HIGH BYTE

0 = ENABLE
1 = DISABLE

0 = NO SHORTING PLUG
X = SHORTING PLUG INSTALLED

8 BIT DATA BUS

LEAVE JUMPER J20 OPEN. (NO SHORTING PLUG)

INSTALL SHORTING PLUGS PER THE FOLLOWING TABLE:

DEVICE DENSITY = 2K BYTES

	JUMPERS PROMS: 1-4		5-8	
	J21	J22	J23	J24
2K	0	X	0	X
4K/8K	X	0	X	0

Figure 2. 2716 or 2816

Table 1. BIP Decoder Switch Settings

Device		2816/2815/2716		2732/2732A	2764
Address Range in Hex (For Full Array)		0000–3FFF	8000–BFFF	0000–7FFF	0000–FFFF
8-Bit Data Bus	SW1	ON	ON	OFF	ON
	SW2	ON	OFF	OFF	OFF
	SW3	ON	OFF	ON	ON
	SW4	ON	ON	ON	ON
	*SW5	ON	ON	ON	ON
16-Bit Data Bus	SW1	OFF	OFF	ON	OFF
	SW2	OFF	OFF	ON	ON
	SW3	ON	OFF	OFF	OFF
	SW4	ON	ON	ON	ON
	*SW5	ON	ON	ON	ON

OFF = No shorting plug.
ON = Install shorting plug.
*Must Be "ON" for 3628A.

SYSTEM ADDRESS A0-15 HEX	DECODER CIRCUIT INPUTS BAF	BAE	BAD	BAC	$\overline{BHEN}$	LA0	$\overline{CE}$'S 7	6	5	4	3	2	1	0	BYTE
0 X X X	0	0	0	0	1	0							0	0	L
					1	1									H
1 X X X	0	0	0	1	1	0							0	0	L
					1	1									H
2 X X X	0	0	1	0	1	0					0	0			L
					1	1									H
3 X X X	0	0	1	1	1	0					0	0			L
					1	1									H
4 X X X	0	1	0	0	1	0			0	0					L
					1	1									H
5 X X X	0	1	0	1	1	0			0	0					L
					1	1									H
6 X X X	0	1	1	0	1	0	0	0							L
					1	1									H
7 X X X	0	1	1	1	1	0	0	0							L
					1	1									H
8 X X X	1	0	0	0	1	0									L
					1	1									H
9 X X X	1	0	0	1	1	0									L
					1	1									H
A X X X	1	0	1	0	1	0									L
					1	1									H
B X X X	1	0	1	1	1	0									L
					1	1									H
C X X X	1	1	0	0	1	0									L
					1	1									H
D X X X	1	1	0	1	1	0									L
					1	1									H
E X X X	1	1	1	0	1	0									L
					1	1									H
F X X X	1	1	1	1	1	0									L
					1	1									H
X=HEX DIGITS	A5	A4	A3	A2	A1	A0	0 7	0 6	0 5	0 4	0 3	0 2	0 1	0 0	
	ADDRESS INPUTS 3628A						OUTPUTS								

L = LOW BYTE
H = HIGH BYTE

0 = ENABLE
1 = DISABLE

0 = NO SHORTING PLUG
X = SHORTING PLUG INSTALLED

8 BIT DATA BUS

LEAVE JUMPER J20 OPEN. (NO SHORTING PLUG)

INSTALL SHORTING PLUGS PER THE FOLLOWING TABLE:

DEVICE DENSITY = 4K BYTES

	JUMPERS			
PROMS:	1-4		5-8	
	J21	J22	J23	J24
2K	0	X	0	X
4K/8K	X	0	X	0

Figure 3. 2732 or 2732A

Table 2. MOS PROM Sockets

Device Density	1–4		5–8	
	J21	J22	J23	J24
2K	O	X	O	X
4K or 8K	X	O	X	O

O = No shorting plug.
X = Install shorting plug.

SYSTEM ADDRESS A0-15 HEX	DECODER CIRCUIT INPUTS						CE'S								BYTE
	BAF	BAE	BAD	BAC	BHEN	LA0	7	6	5	4	3	2	1	0	
0 X X X	0	0	0	0	1	0							0		L
					1	1								0	H
1 X X X	0	0	0	1	1	0							0		L
					1	1								0	H
2 X X X	0	0	1	0	1	0							0		L
					1	1								0	H
3 X X X	0	0	1	1	1	0							0		L
					1	1								0	H
4 X X X	0	1	0	0	1	0					0	0			L
					1	1					0				H
5 X X X	0	1	0	1	1	0					0	0			L
					1	1					0				H
6 X X X	0	1	1	0	1	0					0	0			L
					1	1					0				H
7 X X X	0	1	1	1	1	0					0	0			L
					1	1					0				H
8 X X X	1	0	0	0	1	0			0	0					L
					1	1			0						H
9 X X X	1	0	0	1	1	0			0	0					L
					1	1			0						H
A X X X	1	0	1	0	1	0			0	0					L
					1	1			0						H
B X X X	1	0	1	1	1	0			0	0					L
					1	1			0						H
C X X X	1	1	0	0	1	0	0	0							L
					1	1	0								H
D X X X	1	1	0	1	1	0	0	0							L
					1	1	0								H
E X X X	1	1	1	0	1	0	0	0							L
					1	1	0								H
F X X X	1	1	1	1	1	0	0	0							L
					1	1	0								H
X=HEX DIGITS	A5	A4	A3	A2	A1	A0	0 7	0 6	0 5	0 4	0 3	0 2	0 1	0 0	
	ADDRESS INPUTS 3628A						OUTPUTS								

L = LOW BYTE
H = HIGH BYTE

0 = ENABLE
1 = DISABLE

0 = NO SHORTING PLUG
X = SHORTING PLUG INSTALLED

8 BIT DATA BUS

LEAVE JUMPER J20 OPEN. (NO SHORTING PLUG)

INSTALL SHORTING PLUGS PER THE FOLLOWING TABLE:

DEVICE DENSITY = 8K BYTES

	JUMPERS			
PROMS:	1-4		5-8	
	J21	J22	J23	J24
2K	0	X	0	X
4K/8K	X	0	X	0

Figure 4. 2764

SYSTEM ADDRESS A0–15 HEX	DECODER CIRCUIT INPUTS						$\overline{CE}$'S								BYTE
	BAF	BAE	BAD	BAC	$\overline{BHEN}$	LA0	7	6	5	4	3	2	1	0	
0XXX	0	0	0	0	0	0							0	0	W
					1	0							0	0	L
					0	1									H
1XXX	0	0	0	1	0	0					0	0			W
					1	0					0	0			L
					0	1									H
2XXX	0	0	1	0	0	0			0	0					W
					1	0			0	0					L
					0	1									H
3XXX	0	0	1	1	0	0	0	0							W
					1	0	0	0							L
					0	1									H
4XXX	0	1	0	0	0	0									W
					1	0									L
					0	1									H
5XXX	0	1	0	1	0	0									W
					1	0									L
					0	1									H
6XXX	0	1	1	0	0	0									W
					1	0									L
					0	1									H
7XXX	0	1	1	1	0	0									W
					1	0									L
					0	1									H
8XXX	1	0	0	0	0	0									W
					1	0									L
					0	1									H
9XXX	1	0	0	1	0	0									W
					1	0									L
					0	1									H
AXXX	1	0	1	0	0	0									W
					1	0									L
					0	1									H
BXXX	1	0	1	1	0	0									W
					1	0									L
					0	1									H
CXXX	1	1	0	0	0	0									W
					1	0									L
					0	1									H
DXXX	1	1	0	1	0	0									W
					1	0									L
					0	1									H
EXXX	1	1	1	0	0	0									W
					1	0									L
					0	1									H
FXXX	1	1	1	1	0	0									W
					1	0									L
					0	1									H
	A5	A4	A3	A2	A1	A0	07	06	05	04	03	02	01	00	
X = HEX DIGITS	ADDRESS INPUTS 3628A						OUTPUTS								

L = LOW BYTE
H = HIGH BYTE

0 = ENABLE
1 = DISABLE

0 = NO SHORTING PLUG
X = SHORTING PLUG INSTALLED

16 BIT DATA BUS

INSTALL SHORTING PLUG AT JUMPER J20.

INSTALL SHORTING PLUGS PER THE FOLLOWING TABLE:

DEVICE DENSITY = 2K BYTES

	JUMPERS			
	PROMS: 1-4		5-8	
	J21	J22	J23	J24
2K	0	X	0	X
4K/8K	X	0	X	0

Figure 5. 2716 or 2816

SYSTEM ADDRESS A0–15 HEX	DECODER CIRCUIT INPUTS						$\overline{CE}$'S								BYTE
	BAF	BAE	BAD	BAC	$\overline{BHEN}$	LA0	7	6	5	4	3	2	1	0	
0XXX	0	0	0	0	0 1 0	0 0 1							0 0	0 0	W L H
1XXX	0	0	0	1	0 1 0	0 0 1							0 0	0 0	W L H
2XXX	0	0	1	0	0 1 0	0 0 1					0 0	0 0			W L H
3XXX	0	0	1	1	0 1 0	0 0 1					0 0	0 0			W L H
4XXX	0	1	0	0	0 1 0	0 0 1			0 0	0 0					W L H
5XXX	0	1	0	1	0 1 0	0 0 1			0 0	0 0					W L H
6XXX	0	1	1	0	0 1 0	0 0 1	0 0	0 0							W L H
7XXX	0	1	1	1	0 1 0	0 0 1	0 0	0 0							W L H
8XXX	1	0	0	0	0 1 0	0 0 1									W L H
9XXX	1	0	0	1	0 1 0	0 0 1									W L H
AXXX	1	0	1	0	0 1 0	0 0 1									W L H
BXXX	1	0	1	1	0 1 0	0 0 1									W L H
CXXX	1	1	0	0	0 1 0	0 0 1									W L H
DXXX	1	1	0	1	0 1 0	0 0 1									W L H
EXXX	1	1	1	0	0 1 0	0 0 1									W L H
FXXX	1	1	1	1	0 1 0	0 0 1									W L H
	A5	A4	A3	A2	A1	A0	0 7	0 6	0 5	0 4	0 3	0 2	0 1	0 0	
X = HEX DIGITS	ADDRESS INPUTS 3628A						OUTPUTS								

L = LOW BYTE
H = HIGH BYTE

0 = ENABLE
1 = DISABLE

0 = NO SHORTING PLUG
X = SHORTING PLUG INSTALLED

16 BIT DATA BUS

INSTALL SHORTING PLUG AT JUMPER J20.

INSTALL SHORTING PLUGS PER THE FOLLOWING TABLE:

DEVICE DENSITY = 4K BYTES

	JUMPERS			
	PROMS: 1-4		5-8	
	J21	J22	J23	J24
2K	0	X	0	X
4K/8K	X	0	X	0

Figure 6. 2732 or 2732A

SYSTEM ADDRESS A0–15 HEX	DECODER CIRCUIT INPUTS						$\overline{CE}$'S								BYTE
	BAF	BAE	BAD	BAC	$\overline{BHEN}$	LA0	7	6	5	4	3	2	1	0	
0 X X X	0	0	0	0	0 1 0	0 0 1							0 0	0 0	W L H
1 X X X	0	0	0	1	0 1 0	0 0 1							0 0	0 0	W L H
2 X X X	0	0	1	0	0 1 0	0 0 1							0 0	0 0	W L H
3 X X X	0	0	1	1	0 1 0	0 0 1							0 0	0 0	W L H
4 X X X	0	1	0	0	0 1 0	0 0 1					0 0	0 0			W L H
5 X X X	0	1	0	1	0 1 0	0 0 1					0 0	0 0			W L H
6 X X X	0	1	1	0	0 1 0	0 0 1					0 0	0 0			W L H
7 X X X	0	1	1	1	0 1 0	0 0 1					0 0	0 0			W L H
8 X X X	1	0	0	0	0 1 0	0 0 1			0 0	0 0					W L H
9 X X X	1	0	0	1	0 1 0	0 0 1			0 0	0 0					W L H
A X X X	1	0	1	0	0 1 0	0 0 1			0 0	0 0					W L H
B X X X	1	0	1	1	0 1 0	0 0 1			0 0	0 0					W L H
C X X X	1	1	0	0	0 1 0	0 0 1	0 0	0 0							W L H
D X X X	1	1	0	1	0 1 0	0 0 1	0 0	0 0							W L H
E X X X	1	1	1	0	0 1 0	0 0 1	0 0	0 0							W L H
F X X X	1	1	1	1	0 1 0	0 0 1	0 0	0 0							W L H
	A5	A4	A3	A2	A1	A0	0 7	0 6	0 5	0 4	0 3	0 2	0 1	0 0	

X = HEX DIGITS

ADDRESS INPUTS 3628A

OUTPUTS

L = LOW BYTE
H = HIGH BYTE

0 = ENABLE
1 = DISABLE

0 = NO SHORTING PLUG
X = SHORTING PLUG INSTALLED

16 BIT DATA BUS

INSTALL SHORTING PLUG AT JUMPER J20.

INSTALL SHORTING PLUGS PER THE FOLLOWING TABLE:

DEVICE DENSITY = 8K BYTES

	JUMPERS			
PROMS:	1-4		5-8	
	J21	J22	J23	J24
2K	0	X	0	X
4K/8K	X	0	X	0

Figure 7. 2764

Adjusting the V_{PP} Voltage Level

The high level of the V_{PP} pulse must be calibrated to 21V. The 8085A code sequence shown below or an equivalent write routine can be used to generate a series of V_{PP} pulses for initial calibration purposes.

```
        MVI A,
        COUNT       ; COUNT = number required for
                      delay of at least 40 ms.
        OUT 0       ; RESET the board
WRTLP:
        STA 8000H   ; send a write command to the E2
                      Memory board
        LXI B, 0
DLYLP:
        INX B
        CMP B       ; compare B reg with Count
        JNZ DLYLP   ; done yet?
        OUT 0       ; Yes, RESET board and start again
        JMP WRTLP
```

The oscilloscope used should first be calibrated against a known 21.0V DC source. For the above program loop, the board address is 8000H and the RESET I/O address 00H. Remove the shorting plugs from J9-J12. Set switches SW 1-5 to on, off, off, on, on, respectively. Before running any V_{PP} pulse loop on the E^2 Board,

1. Remove all shorting plugs from jumpers J1-J8
2. Remove *all 2816's* from the board. Otherwise the maximum write cycle capacity at the addressed 2816 could be exceeded and the useful device life could be diminished.

$\overline{\text{XACK}}$ Delay

To select R7 use the following table and the tacc value of the slowest device in the PROM Array:

tacc (ns)	XACL delay (ns)	R7 (ohms)
200	250	6K
250	300	7.5K
350	400	12K
450	500	15K
650	700	24K

USER'S OPERATIONAL INSTRUCTIONS

The Multibus Board can be read by simply sending a Memory Read command to the board.

A byte or word write is performed on the system by doing a normal Memory Write Cycle. The Bus CPU will be informed approximately 25ms later via an Interrupt Line that the E^2 Board has completed the write operation. The CPU clears the Interrupt by RESETING the E^2 Board. The board is then ready for a new command.

Each 2816 E^2PROM on the E^2 Board can be erased with one write operation. To accomplish this, the board is first put into Chip Erase Mode by doing an I/O Write cycle to the Chip Erase I/O address. The data written during the I/O Write cycle is not used by the E^2 Board. A Memory Write cycle is then performed on the 2816 to be erased. As with the data write cycle described above, an Interrupt Line will be lowered when the Chip Erase operation has been completed. The CPU issues a RESET to the E^2 Board by doing an I/O Write cycle to the RESET I/O address. (The data put on the system bus during the RESET I/O write cycle is not used.) The 2816 that was written now contains all 1's, the Chip Erase Mode is cleared, and the E^2 Board is ready for the next command.

After powering up the E^2 Memory Board, the CPU must send an initial RESET to the E^2 Board to prepare it for normal operation. This should not occur until the CPU has been running for at least 1 second.

E^2 Memory Board Operation Summary

Operation	CPU Action	E^2 Board Action	Comments
Read	Memory Read command $\overline{MRDC}$	data requested is put on the bus	Minimum delay from $\overline{MRDC}$ to $\overline{XACK}$ = 250 ns
Write	Memory Write command $\overline{MWTC}$		byte or word erase and byte or word write is performed on 2816 or 2816 pair
		an $\overline{INT}$ line is pulled low when operation is done	
	issue RESET		
Chip Erase	Set Chip Erase Mode with an I/O Write Command—then send an $\overline{MWTC}$ to the one or two 2816's to be erased		One or two 2816's are set to all 1's
		$\overline{INT}$ goes low when done	
	issue RESET		
Initialize E^2 Board	After the CPU starts running, wait 1 second, then send RESET command.		E^2 Board is ready for operation

USER PROGRAM EXAMPLE

This sample program transfers a block of data from a RAM buffer to the E^2 memory on the E^2 Memory Board. The system data bus is 8-bits wide, and the RAM memory block is located from 0 to 64K. (The Memory Board is located from 8000H to BFFFH. The system RAM is inhibited by $\overline{INH1}$ from the E^2 Board whenever the latter is accessed.) All 8 MOS PROM sockets are loaded with 2816's. The I/O addresses are 00H for RESET and 01H for Chip Erase Mode. The data transfer in this example consists of less than 4K bytes. The transfer is started by doing an initial Memory Write Cycle to the E^2 Board. Following the Write Cycle, the interrupt subroutine (SRVINT) RESET's the E^2 Board after each write operation has been completed. The subroutine will then take the next data byte from the RAM source block and write it to the next E^2 location. At this point, the subroutine returns control to the main program which in this example idles in a tight loop. An actual system CPU would continue doing its main processing until the present write operation was done. Once the data transfer is started, the interrupt subroutine will take care of reseting the E^2 Board and writing the next byte. The subroutine takes only about 100 μs out of every 20 ms of CPU time to carry out the data transfer. The data transfer is thus highly efficient.

```
LOC  OBJ        LINE       SOURCE STATEMENT

                    1 ;
                    2 ;
                    3 ;*****************************************************************
                    4 ;
                    5 ;
                    6 ;          INTEL CORPORATION
                    7 ;
                    8 ;          SPD E2 MULTIBUS COMPATIBLE MEMORY BOARD
                    9 ;
                   10 ;          APPLICATION PROGRAM
                   11 ;
                   12 ;
                   13 ;
                   14 ;
                   15 ;*****************************************************************
                   16 ;
                   17 ;
                   18 ;
                   19 ;
                   20 ;     THIS PROGRAM DOES A DATA TRANSFER
                   21 ;     BETWEEN A BLOCK OF RAM AND THE E2
                   22 ;     MEMORY BOARD
                   23 ;
                   24 ;     THE RAM BLOCK IS LOCATED AT 0C000H
                   25 ;
                   26 ;     THE E2 BOARD IS LOCATED AT ADDRESS 8000H
                   27 ;
                   28 ;     IO ADDRESSES:
                   29 ;          00H - RESET
                   30 ;          01H - CHIP ERASE
                   31 ;
                   32 ;     INTERRUPT LINE 7 (INT7') IS USED TO INFORM
                   33 ;     THE CPU WHEN THE E2 BOARD HAS COMPLETED A
                   34 ;     BYTE ERASE/WRITE OR A CHIP ERASE COMMAND
                   35 ;
                   36 ;     0E00H DATA BYTES ARE TRANSFERRED
                   37 ;
                   38 ;
                   39 ;
                   40 ;
                   41 ;
8000               42 STRADD          EQU     8000H
00FC               43 INTMSK          EQU     0FCH
00FD               44 OLRSTR          EQU     0FDH
000D               45 CR              EQU     0DH
000A               46 LF              EQU     0AH
0009               47 EXIT            EQU     9
                   48 ;
                   49                 EXTRN   CO,CI,ISIS
                   50 ;
                   51 ;
                   52 ;
7800               53                 ORG     7800H
                   54 ;
```

```
LOC  OBJ          LINE       SOURCE STATEMENT

7800 C34678         55         JMP     INIT
                    56 ;
                    57 ;
                    58 ;
                    59 ;   ************************************************
                    60 ;
                    61 ;
                    62 ;
                    63 ;          DATA STRINGS
                    64 ;
                    65 ;
                    66 ;************************************************
                    67 ;
                    68 ;
7803 5452414E       69 TRNMSG:          DB      'TRANSFER IN PROGRESS',CR,LF,0FFH
7807 53464552
780B 20494E20
780F 50524F47
7813 52455353
7817 0D
7818 0A
7819 FF
781A 44415441       70 ERRMSG:          DB      'DATA ERROR',CR,LF,0FFH
781E 20455252
7822 4F52
7824 0D
7825 0A
7826 FF
7827 5452414E       71 FINMSG:          DB      'TRANSFER COMPLETE',CR,LF,0FFH
782B 53464552
782F 20434F4D
7833 504C4554
7837 45
7838 0D
7839 0A
783A FF
783B 3D78           72 EBLK:            DW      ESTAT
                    73 ;
                    74 ;
                    75 ;
                    76 ;****************************************
                    77 ;
                    78 ;
                    79 ;     TEMPORARY STORAGE SPACE
                    80 ;
                    81 ;
                    82 ;****************************************
                    83 ;
                    84 ;
783D                85 ESTAT:           DS      2
783F                86 E2BUSY:          DS      1
7840                87 MSGADD:          DS      2
7842                88 SRCADD:          DS      2
7844                89 DESADD:          DS      2
                    90 ;
```

```
LOC  OBJ        LINE       SOURCE STATEMENT

                  91 ;
                  92 ;
                  93 ;
                  94 INIT:
7846 31007E       95         LXI     SP,7E00H
7849 D300         96         OUT     0                ;RESET THE E2 BOARD
                  97 ;
                  98 ;
                  99 ;     LOAD LOCATION 038H(INTERRUPT 7 VECTOR)
                 100 ;     WITH JUMP TO SERVICE INTERRUPT SUBROUTINE
                 101 ;
784B 3EC3        102         MVI     A,0C3H
784D 323800      103         STA     38H
7850 21AE78      104         LXI     H,SRVINT
7853 223900      105         SHLD    39H
                 106 ;
                 107 ;     SET THE SYSTEM INTERRUPT MASK TO
                 108 ;     ENABLE INTERRUPTS 1 AND 7
                 109 ;
                 110 ;
                 111 ;
7856 3E7E        112         MVI     A,7EH
7858 D3FC        113         OUT     INTMSK
785A FB          114         EI
785B 210378      115         LXI     H,TRNMSG              ;INFORM OPERATOR
                 116                                       ;THAT THE TRANSFER
                 117                                       ;HAS BEGUN
                 118 ;
785E 224078      119         SHLD    MSGADD
7861 CDE578      120         CALL    DISMSG
                 121 ;
                 122 ;
                 123 ;
                 124 ;**************************
                 125 ;
                 126 ;     MAIN PROGRAM
                 127 ;
                 128 ;
                 129 MAINPG:
                 130 ;
                 131 ;     LOAD SOURCE RAM BLOCK WITH 55H
                 132 ;
7864 2100C0      133         LXI     H,RAMBLK
                 134 LDLP:
7867 3655        135         MVI     M,55H
7869 23          136         INX     H
786A 3ECE        137         MVI     A,0CEH
786C BC          138         CMP     H
786D C26778      139         JNZ     LDLP
                 140 ;
                 141 ;
7870 3EFF        142         MVI     A,0FFH
7872 77          143         MOV     M,A               ;INSERT END OF DATA STRING
                 144                                   ;MARKER
                 145 ;
```

```
LOC  OBJ        LINE       SOURCE STATEMENT

                  146 ;
                  147 ;     INITIATE DATA TRANSFER
                  148 ;
                  149 ;
7873 210080       150         LXI     H,STRADD
7876 224478       151         SHLD    DESADD          ;STORE STARTING ADDRESS OF
                  152                                 ;E2 BOARD
7879 2100C0       153         LXI     H,RAMBLK
787C 224278       154         SHLD    SRCADD          ;STORE STARTING ADDRESS OF
                  155                                 ;SOURCE RAM BLOCK
787F 7E           156         MOV     A,M             ;FETCH FIRST BYTE FROM SOURCE
7880 2A4478       157         LHLD    DESADD          ;LOAD DESTINATION ADDRESS
                  158                                 ;(E2 BOARD)
                  159 ;
                  160 STXFER:
7883 77           161         MOV     M,A             ;WRITE FIRST BYTE TO E2
                  162                                 ;BOARD
7884 3E01         163         MVI     A,1
7886 323F78       164         STA     E2BUSY          ;SET FLAG INDICATING THAT A
                  165                                 ;DATA TRANSFER IS IN PROGRESS
                  166 ;
                  167 ;
                  168 ;
                  169 ;
                  170 ;       THIS TIGHT LOOP SIMULATES AN ACTUAL SYSTEM CPU
                  171 ;       DOING ITS MAIN PROCESSING
                  172 ;
                  173 ;
                  174 PROCES:
7889 3A3F78       175         LDA     E2BUSY
788C FE01         176         CPI     1
788E CA8978       177         JZ      PROCES
                  178 ;
                  179 ;
                  180 ;
                  181 ;     ONCE THE DATA TRANSFER IS COMPLETED THE OPERATOR
                  182 ;     IS INFORMED AND CONTROL IS RETURNED TO ISIS
                  183 ;
                  184 ;
                  185 ;
                  186 ;
                  187 FINISH:
7891 212778       188         LXI     H,FINMSG
7894 224078       189         SHLD    MSGADD
7897 CDE578       190         CALL    DISMSG
                  191 ;
                  192 ;
                  193 ;
                  194 RTISIS:
789A 0E09         195         MVI     C,EXIT
789C 113B78       196         LXI     D,EBLK
789F CD0000   E   197         CALL    ISIS
                  198 ;
                  199 ;
                  200 ;
```

```
LOC  OBJ        LINE       SOURCE STATEMENT

                201 ERROR:
78A2 211A78     202        LXI     H,ERRMSG
78A5 224078     203        SHLD    MSGADD
78A8 CDE578     204        CALL    DISMSG
78AB C39A78     205        JMP     RTISIS
                206 ;
                207 ;
                208 ;
                209 ;*****************************
                210 ;
                211 ;
                212 ;      SUBROUTINES
                213 ;
                214 ;
                215 ;
                216 ;
                217 ;
                218 ;      WHEN THE E2 BOARD HAS COMPLETED ITS BYTE
                219 ;      ERASE/WRITE CYCLE THIS ROUTINE WILL VERIFY
                220 ;      THE WRITTEN DATA.  THE NEXT BYTE IS FETCHED
                221 ;      FROM THE RAM BLOCK AND WRITTEN TO THE NEXT
                222 ;      LOCATION ON THE E2 BOARD.
                223 ;
                224 ;
                225 ;
                226 ;
                227 SRVINT:
78AE FB         228        EI
                229 ;
                230 ;      SAVE ALL REG'S
                231 ;
78AF F5         232        PUSH    PSW
78B0 C5         233        PUSH    B
78B1 D5         234        PUSH    D
78B2 E5         235        PUSH    H
78B3 D300       236        OUT     0               ;RESET THE E2 BOARD
                237 ;
78B5 2A4478     238        LHLD    DESADD
78B8 7E         239        MOV     A,M
78B9 2A4278     240        LHLD    SRCADD
78BC BE         241        CMP     M               ;CORRECT DATA?
78BD C2A278     242        JNZ     ERROR
                243 ;
                244 ;      YES, CONTINUE
                245 ;
78C0 23         246        INX     H
78C1 7E         247        MOV     A,M
78C2 FEFF       248        CPI     0FFH            ;END OF STRING MARKER?
78C4 CAD578     249        JZ      DONE
                250 ;
                251 ;      NO, CONTINUE
                252 ;
78C7 224278     253        SHLD    SRCADD
78CA 2A4478     254        LHLD    DESADD
78CD 23         255        INX     H
```

```
LOC  OBJ        LINE       SOURCE STATEMENT

78CE 77          256        MOV     M,A             ;WRITE NEW BYTE TO E2 BOARD
78CF 224478      257        SHLD    DESADD
78D2 C3DA78      258        JMP     REST            ;RESTORE REG'S AND RETURN
                 259 ;
                 260 ;    CLEAR FLAG TO INFORM CPU THAT THE DATA TRANSFER
                 261 ;    IS FINISHED
                 262 ;
                 263 DONE:
78D5 3E00        264        MVI     A,0
78D7 323F78      265        STA     E2BUSY
                 266 ;
                 267 REST:
78DA E1          268        POP     H
78DB D1          269        POP     D
78DC C1          270        POP     B
78DD F3          271        DI
78DE 3E20        272        MVI     A,20H
78E0 D3FD        273        OUT     OLRSTR
78E2 F1          274        POP     PSW
78E3 FB          275        EI
78E4 C9          276        RET
                 277 ;
                 278 ;
                 279 ;
                 280 ;
                 281 ;
                 282 ;
                 283 DISMSG:
78E5 2A4078      284        LHLD    MSGADD
                 285 MSGLP:
78E8 4E          286        MOV     C,M
78E9 3EFF        287        MVI     A,0FFH
78EB BE          288        CMP     M
78EC C8          289        RZ
78ED CD0000   E  290        CALL    CO
78F0 23          291        INX     H
78F1 C3E878      292        JMP     MSGLP
                 293 ;
                 294 ;
                 295 ;
                 296 ;    RAM BLOCK
C000             297        ORG     0C000H
C000             298 RAMBLK:        DS      1000H
                 299 ;
                 300 ;
                 301 ;
                 302 ;
7846             303        END     INIT


PUBLIC SYMBOLS


EXTERNAL SYMBOLS
CI     E 0000    CO     E 0000    ISIS   E 0000
```

```
USER SYMBOLS
CI     E 0000   CO     E 0000   CR     A 000D   DESADD A 7844   DISMSG A 78E5   DONE   A 78D5   E2BUSY A 783F
EBLK   A 783B   ERRMSG A 781A   ERROR  A 78A2   ESTAT  A 783D   EXIT   A 0009   FINISH A 7891   FINMSG A 7827
INIT   A 7846   INTMSK A 00FC   ISIS   E 0000   LDLP   A 7867   LF     A 000A   MAINPG A 7864   MSGADD A 7840
MSGLP  A 78E8   OLRSTR A 00FD   PROCES A 7889   RAMBLK A C000   REST   A 78DA   RTISIS A 789A   SRCADD A 7842
SRVINT A 78AE   STRADD A 8000   STXFER A 7883   TRNMSG A 7803

ASSEMBLY COMPLETE,  NO ERRORS
```

16-BIT DATA BUS STRUCTURE

The Multibus card can use either an 8-bit or 16-bit data bus. The PROM Array is organized in pairs of 8-bit wide MOS PROM's to enable the formation of a 16-bit word. For a 16-bit data bus, the upper byte MOS PROM is enabled whenever $\overline{\text{BHE}}$ (Byte High Enable) is low. The lower byte PROM is enabled when A0 is low. The upper and lower PROM's can be enabled and accessed separately as individual bytes or together to form a word. Accessing data by words takes half the time required to do byte operations; thus the advantage of 16-bit systems over 8-bit systems.

HARDWARE DESCRIPTION

Overview

This discussion assumes an 8-bit data bus is being used and applies equally to a 16-bit-wide system except that whenever a byte-wide operation is being described, two bytes (two MOS PROMs) in parallel are being affected.

The E^2 Board hardware consists of the following sections:

1. Sequencing and Timing
2. $\overline{\text{XACK}}$ Generation
3. Bus Address Decoding
4. PROM Array Decoding
5. Data and Address Latches and Buffers
6. V_{PP} and $\overline{\text{OE}}$ Drivers
7. 5V to 24V Converter
8. Write Protection Circuitry

See the block diagram in Figure 8.

A brief description of the function of each circuit block will be given. The circuit operation will then be discussed in detail in the subsections to follow.

The Sequencing and Timing circuitry generates the signals necessary to do the byte erase, byte write, chip erase cycles, and read cycles on the E^2 PROMs.

The $\overline{\text{XACK}}$ Generator returns the $\overline{\text{Transfer Acknowledge}}$ signal to the Multibus Bus Master after receiving any memory or I/O command.

The bus Address Decoder performs 2 functions:

1. Enabling the E^2 Board within its assigned address block for memory operations.
2. Enabling the RESET function and setting the Chip Erase Mode whenever the proper I/O addresses are written to.

The PROM Array Decoder enables the proper MOS PROM for any given memory address.

The Address and Data Latches hold the Bus address and data values for the duration of the E^2 write and Chip Erase operations. During read operations, the Data Buffers transfer the accessed data to the Multibus.

The V_{PP} and $\overline{\text{OE}}$ Drivers provide the high voltage pulses required for the byte erase, byte write, and Chip Erase cycles while the 5V to 24V Converters provide the supply voltage for the V_{PP} and $\overline{\text{OE}}$ drivers. Tied to these circuits is the Write Protection Circuitry which prevents any spurious write cycles from occurring during the system power up and power down transitions.

The figures referenced in the following subsections are shown in the Schematic Section.

Sequencing and Timing

READ OPERATION

When $\overline{\text{MRDC}}$ goes low the $\overline{\text{RDEN}}$ signal also goes low. $\overline{\text{WR Mode}}$ is normally high when the E^2 Board is not being accessed and is not performing an operation. $\overline{\text{WR Mode}}$ also stays high during a Read operation. The address latches in Figure 12 remain transparent, and lines BAC–BAF select one of the MOS PROMs in the PROM Array via the 3628A Bipolar PROM decoder. In Figure 13 the $\overline{\text{RDEN}}$ signal enables the 8287 transceivers to gate the output data onto the multibus. $\overline{\text{RDEN}}$ also causes $\overline{\text{OE}}$ to go low to read the data out of the MOS PROMs.

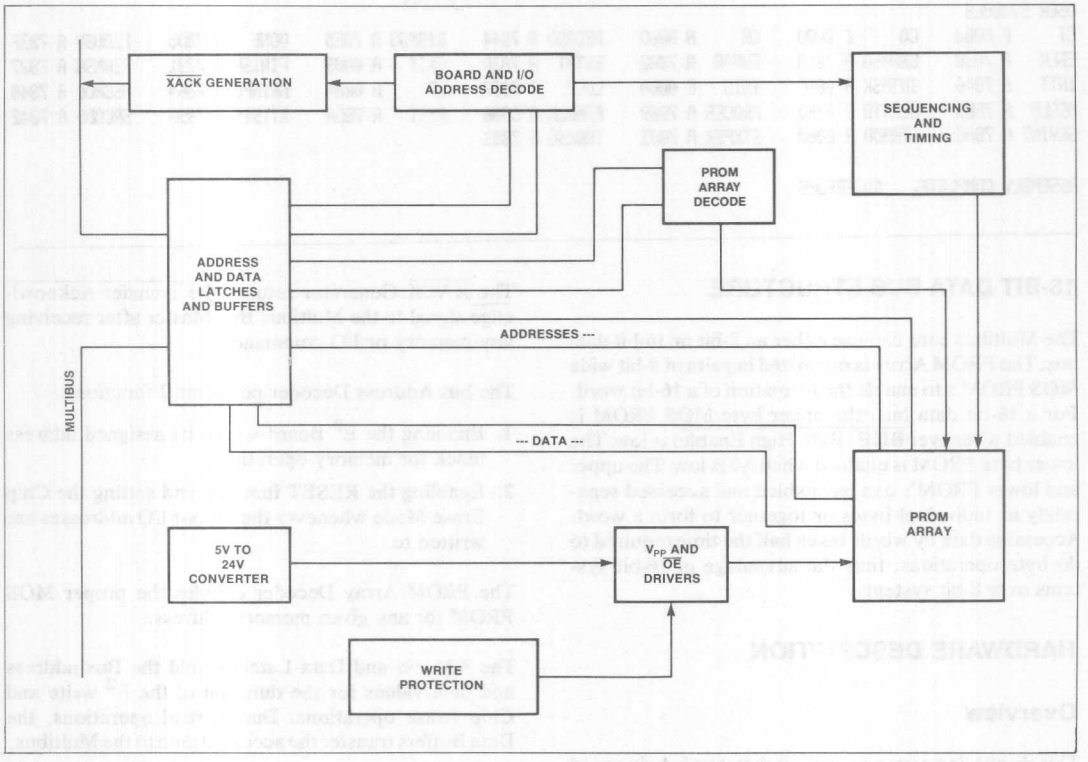


Figure 8. E^2 Board Block Diagram

WRITE OPERATION

Refer to Figures 10 and 11 and to the timing diagram in Figure 9.

When $\overline{MWTC}$ goes low the $\overline{BD\ WR}$ signal also goes low which sets the $\overline{WR}$ Mode FF. The falling edge of $\overline{WR\ Mode}$ latches the addresses as the falling edge of $\overline{BD\ WR}$ latches the data. The rising edge of BD WR sets the Start Erase Cycle FF, which in turn starts the one-shot chain.

The first one-shot delays the rising edge of V_{PP} to provide some set-up time for $\overline{CE}$. When the delay one-shot times out, it triggers the V_{PP} ON one-shot. This second one-shot turns on the V_{PP} driver for approximately 10ms. This is the byte erase cycle. Although the bus data has been latched, the latch outputs are not yet enabled by $\overline{Data\ In\ En}$ (Figure 13). The 1K pullup resistors on the LD0–LD15 lines pull the high-impedance latch outputs up to 5V. As a result, the data inputs to the E^2PROM are all 1's which in turn erase the addressed data byte. When the V_{PP} ON one-shot times out, the V_{PP} Discharge one-shot is triggered. The V_{PP} driver is now shut off ($\overline{V_{PP}\ ON}$ = 1) but the voltage on the V_{PP} line will take a long time to discharge to 5V. This is due to the 4.7 uf low-frequency decoupling capacitor, connected from V_{PP} to ground; the capacitor is needed to suppress low frequency noise (See Figure 17). In order to pull the V_{PP} line down fast, the V_{PP} Discharge signal turns on Q8 which discharges the capacitor. When the third one-shot times out, the Cycle Done One-Shot starts, which clears the start Erase Cycle FF and forces $\overline{CE}$ high (Figure 16 and Figure 10). The rising edge of Cycle Done sets the Start Write Cycle FF and causes the 74LS393 counter (Figure 10) to increment from 0 to 1. This starts the one-shot chain again to perform the byte write cycle. The $\overline{Data\ In\ En}$ signal enables the latched data onto the input lines. The one-shot chain then delays, activates V_{PP}, and discharges the V_{PP} line. This time the data byte is written into the selected 2816 address location. The 74LS393 counter is incremented a second time and its QB output lowers one of the Multibus Interrupt Lines. The CPU RESETS the E^2 board with an $\overline{IOWC}$ command. The WR Mode FF, the Start Write FF and the 74LS393 counter are cleared, and the E^2 Board is ready for the next operation.

CHIP ERASE OPERATION

The Chip Erase Operation is quite similar to the byte erase operation. The differences are:

1. The Chip Erase FF is set by the Multibus CPU before initiating the write operation. The Chip Erase FF is set by doing an $\overline{\text{IOWC}}$ command to the Chip Erase address.
2. When the write operation begins, the $\overline{\text{OE}}$ signal is raised to 14.5V (Figure 17). The byte erase cycle proceeds as before.
3. At the end of the byte erase cycle the Cycle Done signal does not set the Start Write Cycle FF. Instead, the Start Write Cycle FF is held in a clear state by the $\overline{\text{INH}}$ Byte Write signal shown in Figure 10. Cycle Done increments the 74LS393 counter from 0 to 1. The Q_A output is now used to lower one of the Interrupt Lines to signal the CPU that the Chip Erase Operation is complete. When the CPU resets the E^2 Board, the Chip Erase FF is also cleared.

INITIALIZATION

The E^2 Board must be RESET after power up. Due to the write protection circuitry delay period after power up, the RESET should not be sent until at least 1 second after the CPU starts running. Once the board is RESET, it is ready for a command.

$\overline{\text{XACK}}$ Generation

(Figure 11) The $\overline{\text{XACK}}$ ($\overline{\text{Transfer}}$ $\overline{\text{Acknowledge}}$) signal is driven low after a delay period determined by the t_{ACC} of the slowest MOS PROM in the PROM Array. $\overline{\text{XACK}}$ stays low until the Memory or I/O command goes back high. See the XACK delay subsection.

Bus Address Decoding

Two sets of addresses need to be decoded for the E^2 Memory Board: the memory space address for the PROM Array and the I/O address for Chip Erase Mode and the RESET function. The 74LS85 comparators in Figure 16, along with Jumpers J9–J12 and J13–J19, are used to select the desired addresses and generate the appropriate enabling signals when the selected addresses appear on the Multibus. The board memory address is determined by jumpers J9–J12. When the correct memory address is put on the bus, the $\overline{\text{MEM}}$ $\overline{\text{EN}}$ signal goes low. The $\overline{\text{INH1}}$ ($\overline{\text{INHIBIT}}$ $\overline{\text{RAM}}$) signal is driven low on the Multibus to disable any RAM memory that is occupying the same memory space as the E^2 Board.

When the selected I/O address appears on the system bus and $\overline{\text{IOWC}}$ goes low the $\overline{\text{BD}}$ $\overline{\text{IOW}}$ signal goes low (Figure 16). If ADR0 is low, the RESET function is activated. If ADR0 is high, the Chip Erase FF is set.

MOS PROM Array Decoder

(Figure 12) Details on how to program the 3628A Bipolar PROM decoder are given in the Installation Instructions under PROM Array Decoder. The 3628A acts as a sophisticated decoder. The address input to the 3628A allows a maximum address range of 64K with a minimum resolution of 2K. The A0 and BHEN input signals enable the 3628A to select the lower byte MOS PROM, the upper byte MOS PROM, or both in parallel. The 3628A output lines connect to an 8282 8-bit latch. The 8282 latches the decoder output to provide $\overline{\text{CE}}$ for the 10ms erase and write cycles. During read cycles the 8282 simply acts as a buffer for the decoder.

Address and Data Latches and Buffers

The 8283s and 8282 of Figures 12 and 13 latch the address and data from the Multibus when the CPU issues an $\overline{\text{MWTC}}$ command. These address and data values stay latched throughout the Write Operation. The addresses are latched on the falling edge of $\overline{\text{WR}}$ Mode. The input data from the bus is latched by the falling edge of $\overline{\text{WR}}$ EN (See Figure 13). For Read operations the $\overline{\text{WR}}$ Mode signal stays high. A high on the 8283 STB input puts these latches in "transparent" mode: they act as buffers for the bus addresses. The 8287s in Figure 13 act as data output buffers for the accessed MOS PROM data.

The 8286 is used when the system data bus is 8-bits wide. This transceiver transfers the data byte from the LD0–LD7 low byte data bus to LD8–LD15 upper byte data bus for write operations. For read operations, the data byte from the LD8–LD15 byte bus is transferred to the LD0–LD7 data byte bus. This byte swapping circuit is actually adapting the 16-bit upper/lower byte structure to an external 8-bit wide data bus. See the section on 16-bit Data Bus Structure for more information.

V_{PP} and $\overline{\text{OE}}$ Drivers

Refer to Figure 17. The V_{PP} driver provides the 21V V_{PP} programming pulse for the 2816. The pulse goes from 5V to 21V with an exponential rising edge. The $\overline{\text{OE}}$ driver is used to provide nominal TTL levels for read and write operations. This driver also provides a 14.5V level for the Chip Erase cycle.

5V to 24V Converter

Refer to Applications Note AP-103 in the E^2PROM Applications Handbook. (Also Figure 18.)

Write Protection Circuitry

The Write Protection circuits are designed to prevent the TTL control logic from causing an E^2 write/erase cycle to occur during the periods of board power up or power down. The 747 op-amp in Figure 18 senses when the board 5V supply has dropped below the voltage level on C41. When this happens, the op-amp disables the V_{PP} driver by grounding C38 (Figure 17). This prevents the capacitor from charging up the 21V—it is the exponential rising voltage on this capacitor which is used to generate the V_{PP} programming pulse's rising edge.

When the E^2 Board is powered up, the rising V_{CC} voltage begins charging up capacitor C30. Until the voltage on C30 is high enough to turn on Q4 (approximately 3.5V), this transistor will hold the $\overline{V_{PP}\ OFF}$ signal low. This is the same signal in Figures 18 and 17 that is used by the power-down circuit described above. This keeps the V_{PP} driver disabled until about 500ms after V_{CC} reaches 5V. About 400ms before V_{CC} reaches 5V, the Write Protect FF in Figure 10 is set by the Multibus $\overline{INIT}$ signal. This FF will hold the V_{PP} ON one-shot disabled until the CPU RESETS the E^2 Memory Board and the Write Protect FF. Thus, the Q4/C30 circuit holds the V_{PP} driver off until long after the TTL logic has stabilized and the Write Protect FF has disabled the V_{PP} ON one-shot.

The purpose of the Write Protect Flip Flop is to prevent the 2816s from being written to by the Intellec Monitor Program immediately after power up. The Intellec Monitor tries to write to every location in its addressable range after power up. This is done to determine how much RAM is in the system. In a non-Intellec system the $\overline{INIT}$ signal is not really needed as long as no System Write commands occur other than those generated by the user.

SCHEMATIC DIAGRAMS

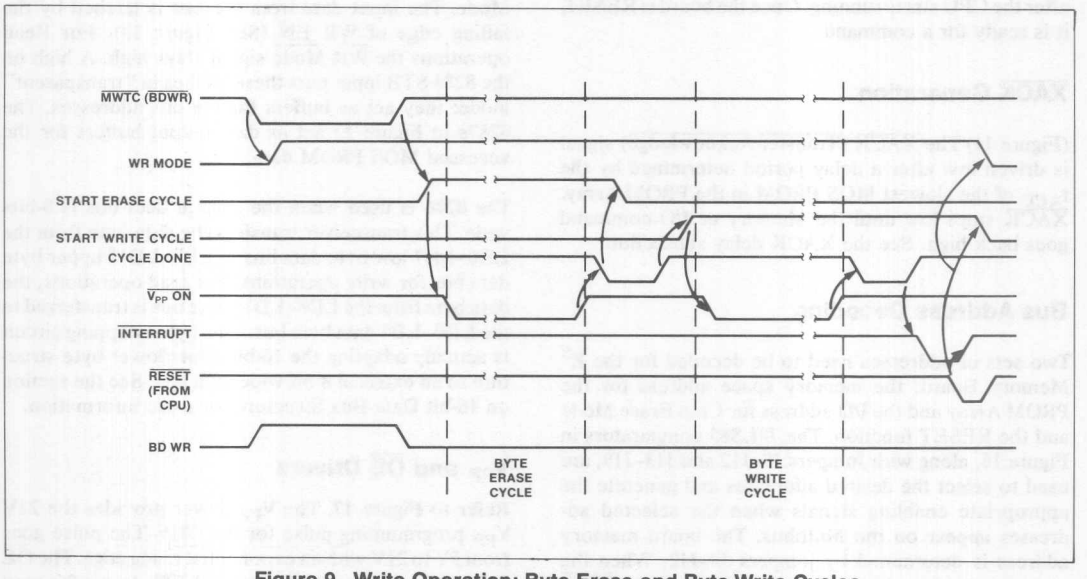


Figure 9. Write Operation: Byte Erase and Byte Write Cycles

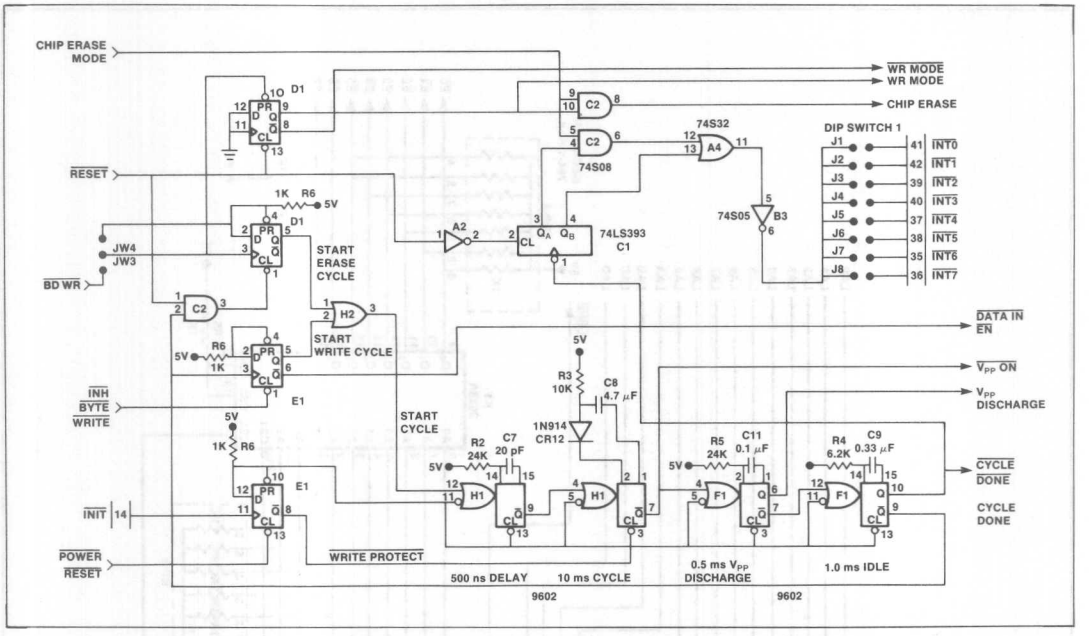
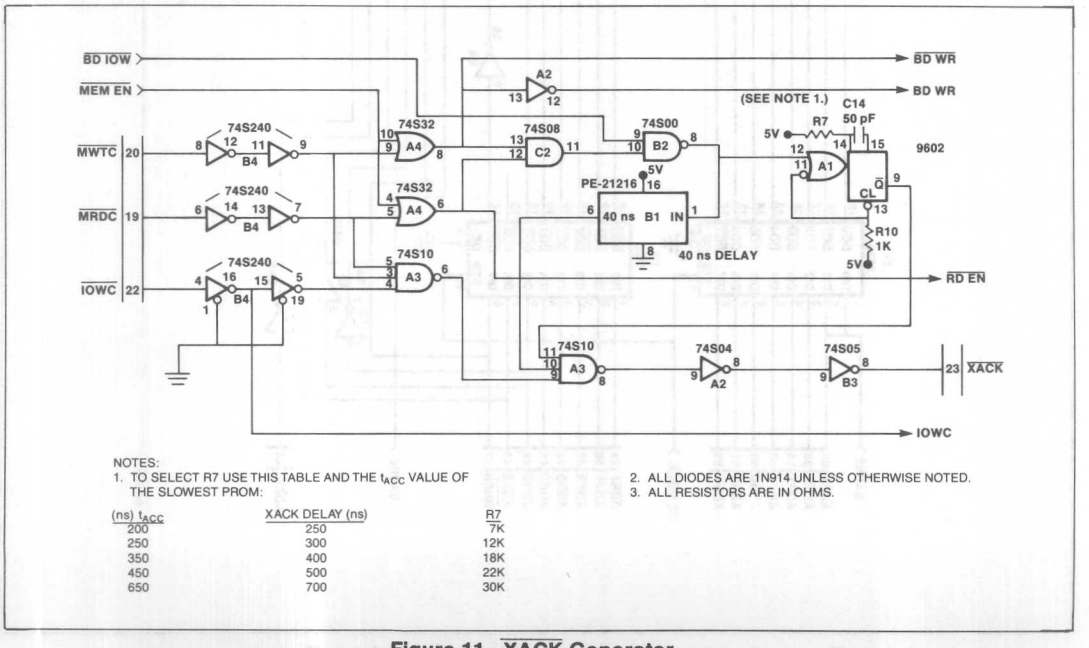


Figure 10. Write and Erase Sequencing and Timing



NOTES:
1. TO SELECT R7 USE THIS TABLE AND THE t_{ACC} VALUE OF THE SLOWEST PROM:

(ns) t_{ACC}	XACK DELAY (ns)	R7
200	250	7K
250	300	12K
350	400	18K
450	500	22K
650	700	30K

2. ALL DIODES ARE 1N914 UNLESS OTHERWISE NOTED.
3. ALL RESISTORS ARE IN OHMS.

Figure 11. $\overline{\text{XACK}}$ Generator

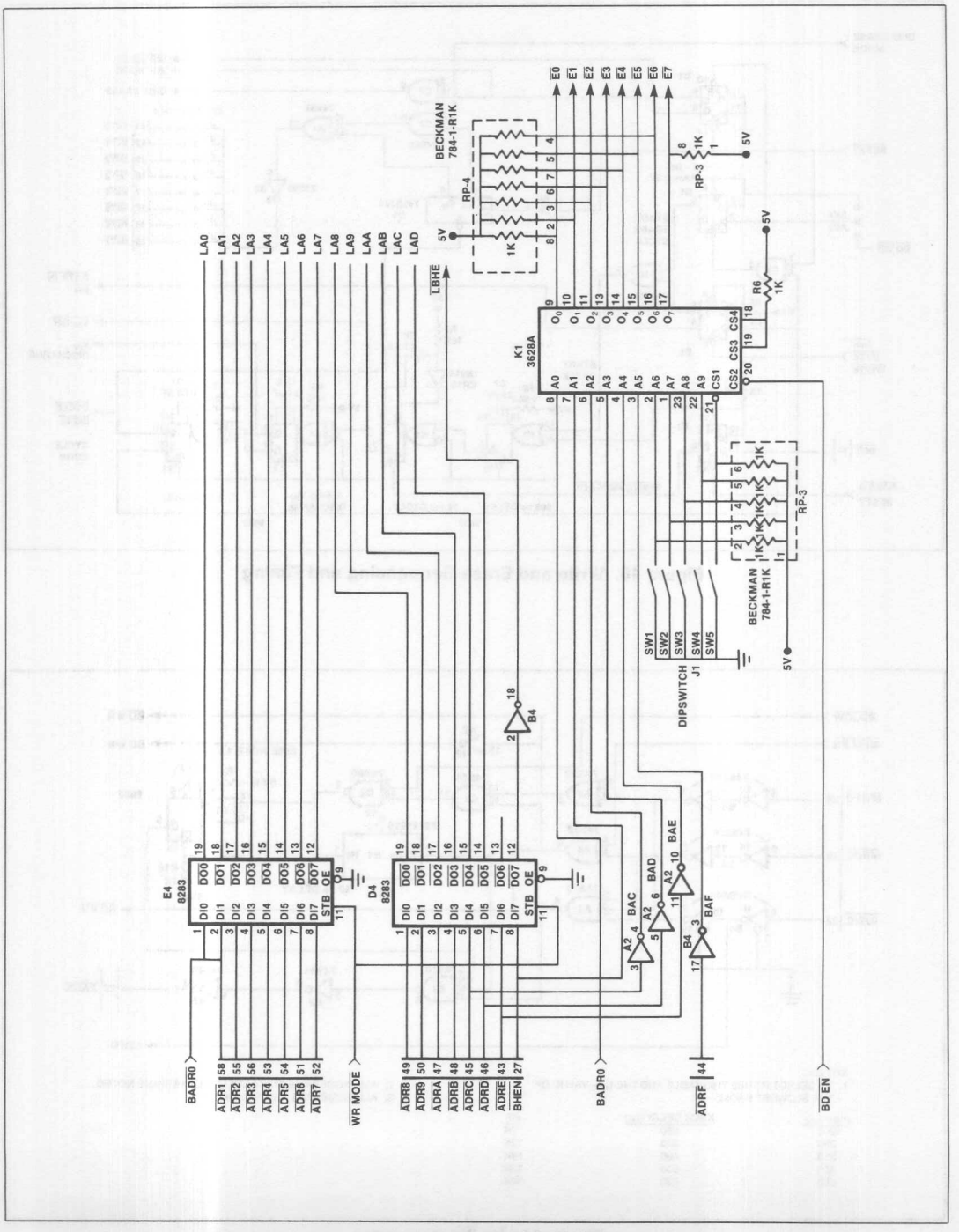


Figure 12. Address Latches and MOS PROM Array Decoder

AFN-02067A

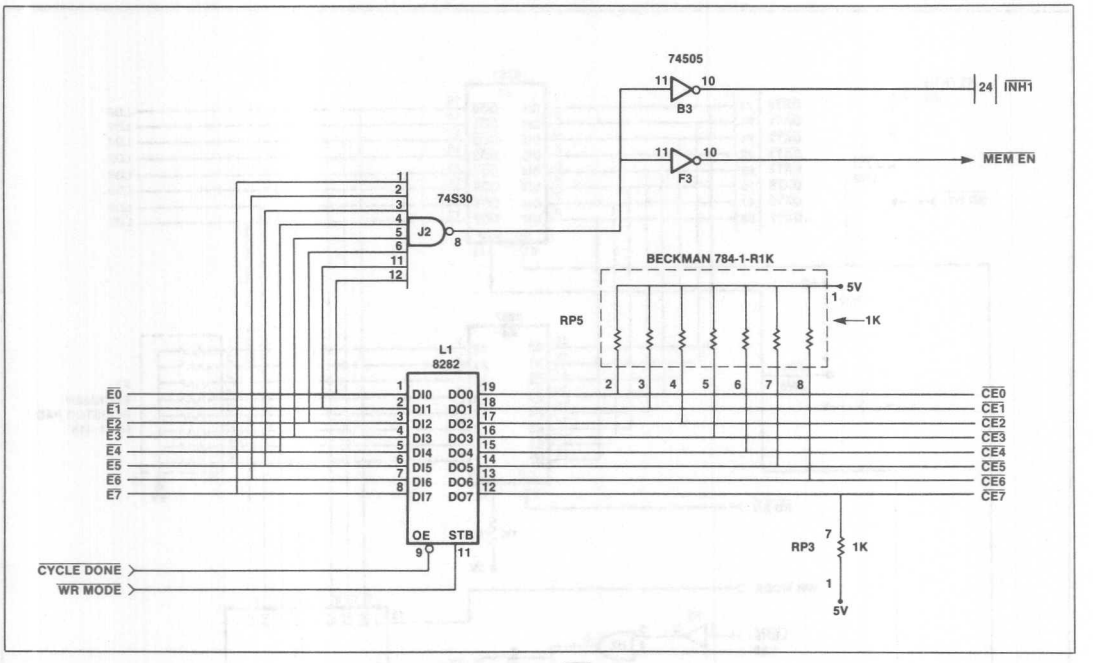


Figure 12a. $\overline{CE}$ Latch

AFN-02067A

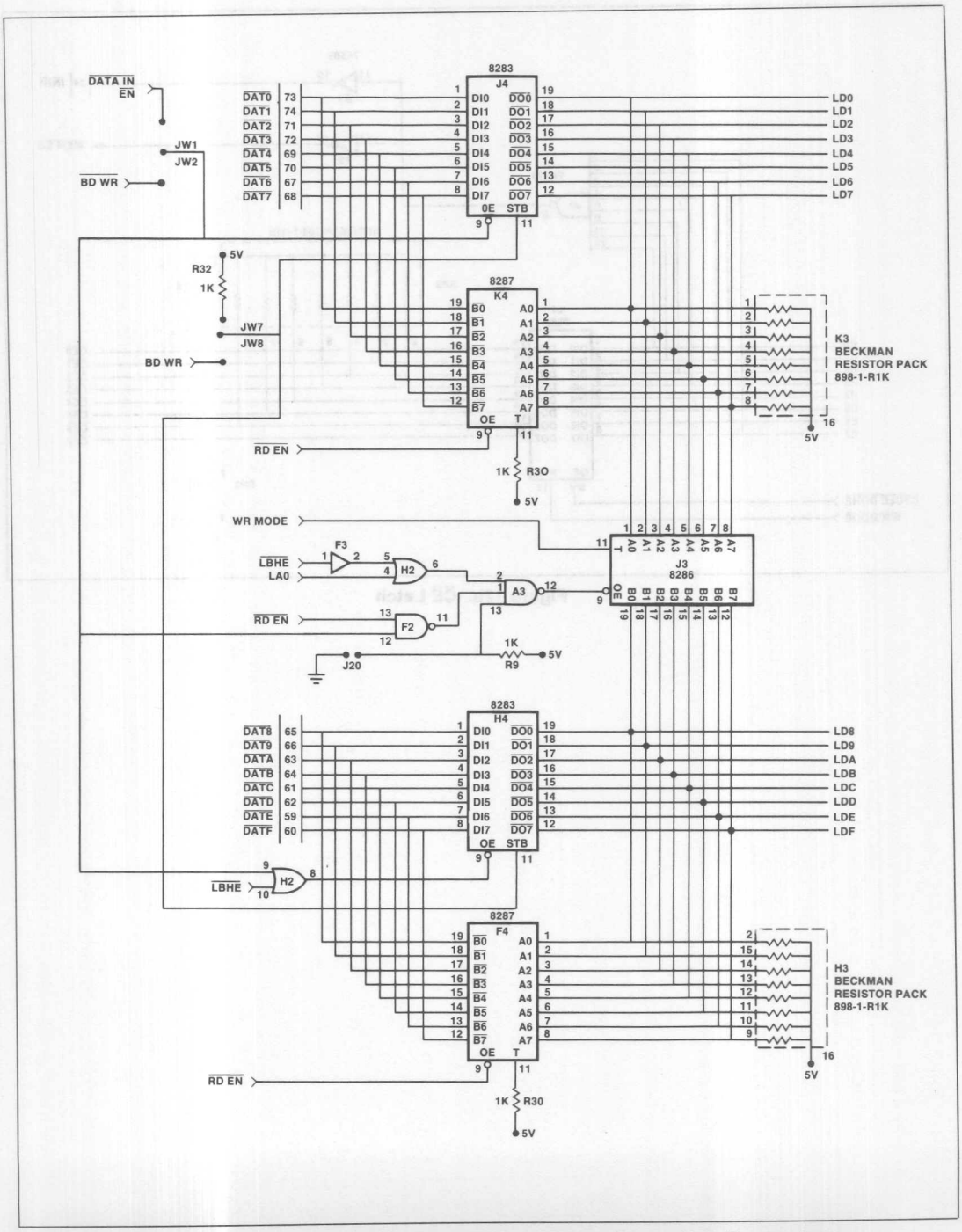


Figure 13. Data In Latch, Data Out Buffer, and Upper/Lower Byte Transceiver

AFN-02067A

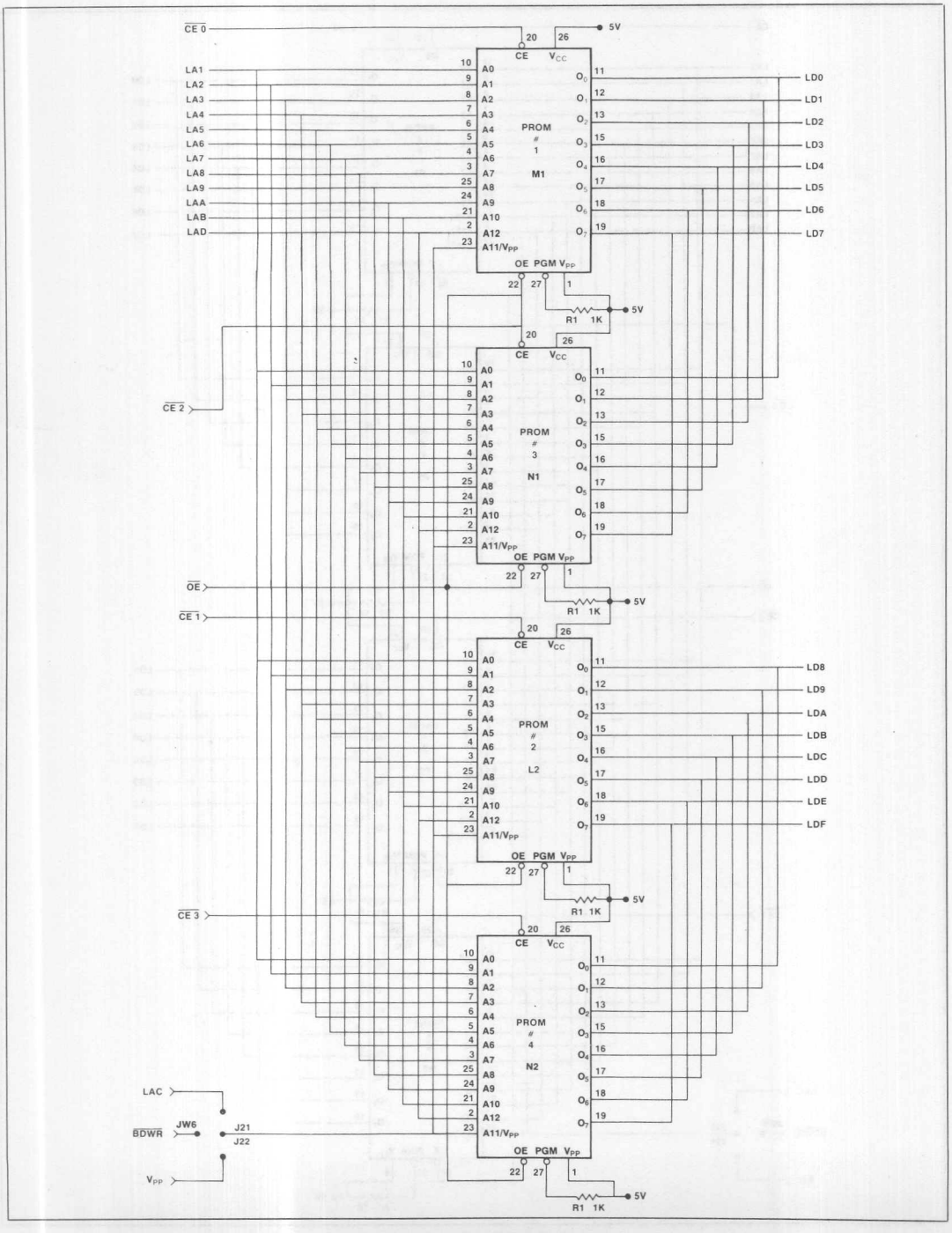


Figure 14. MOS PROM Array

AFN-02067A

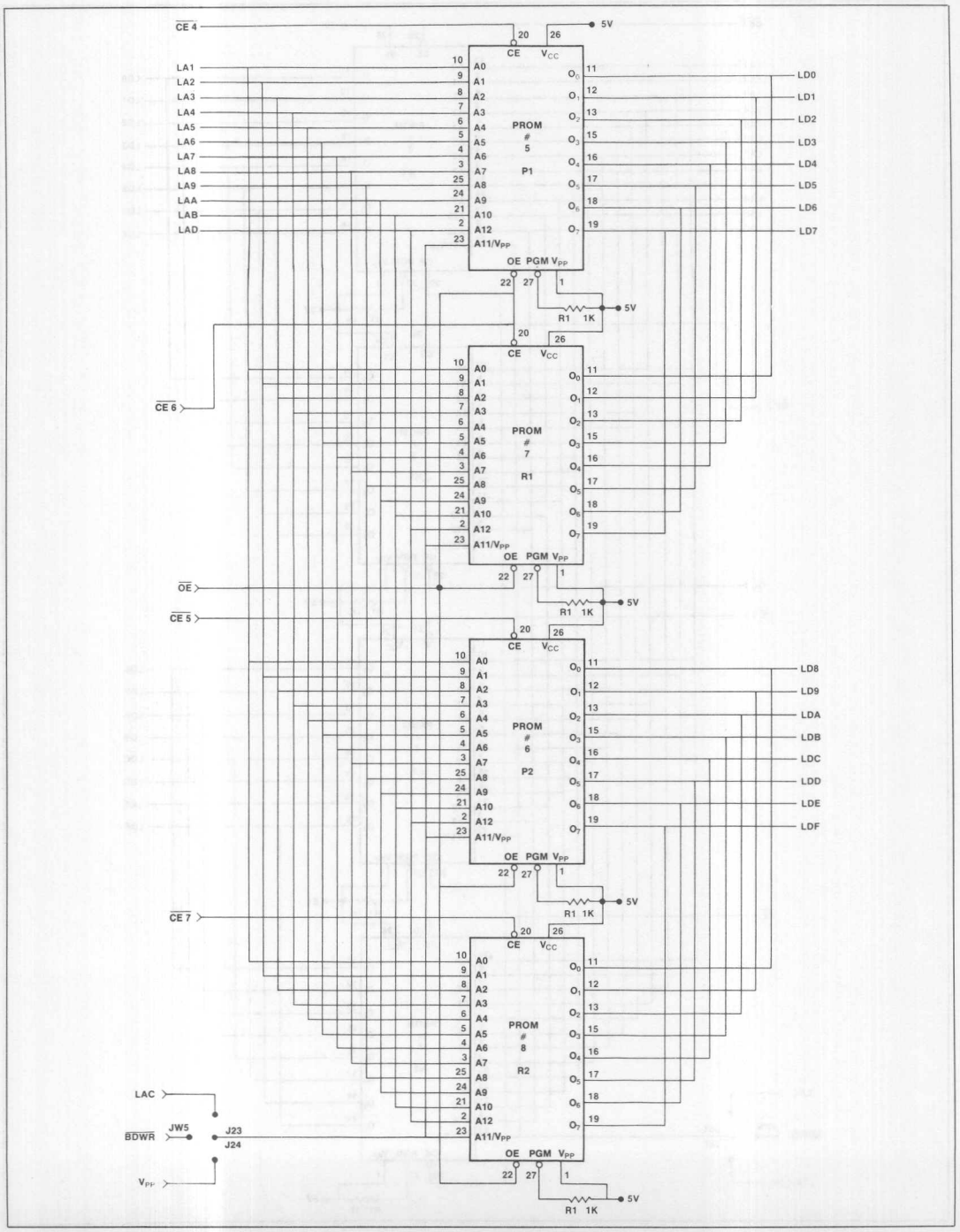


Figure 15. MOS PROM Array

AFN-02067A

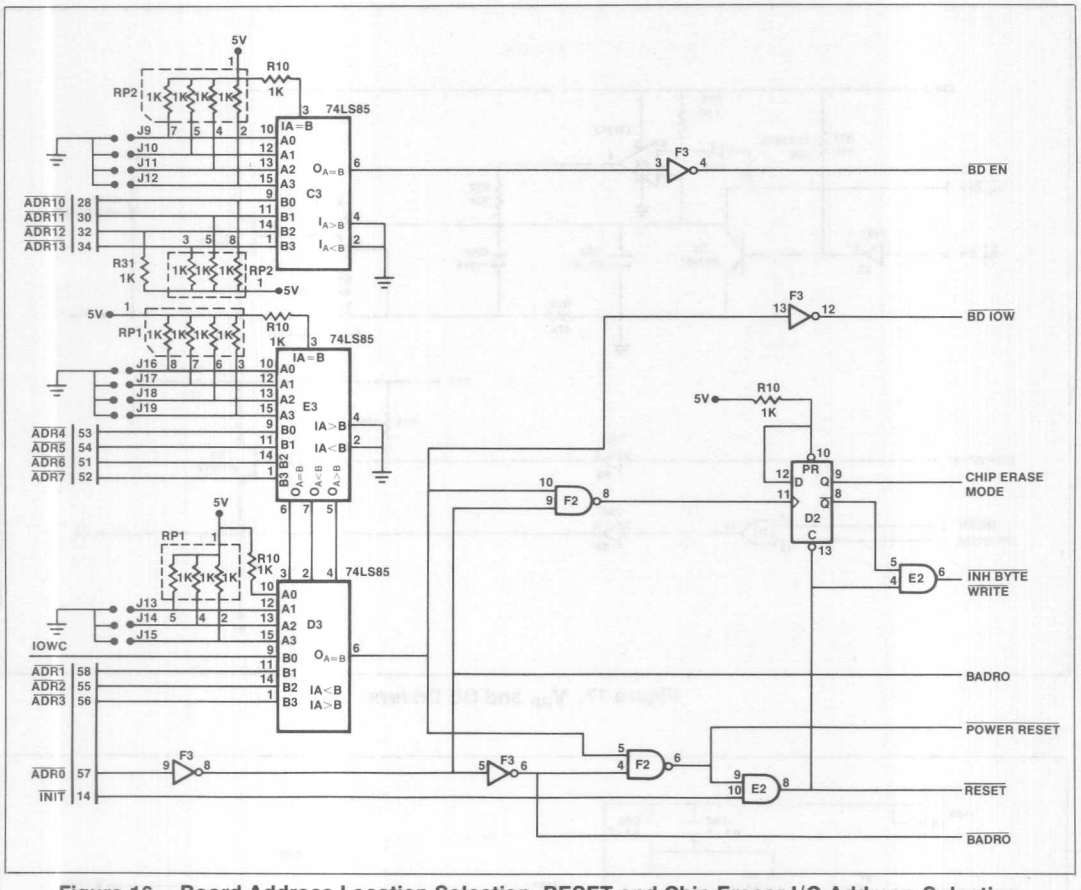


Figure 16. Board Address Location Selection, RESET and Chip Eraser I/O Address Selection

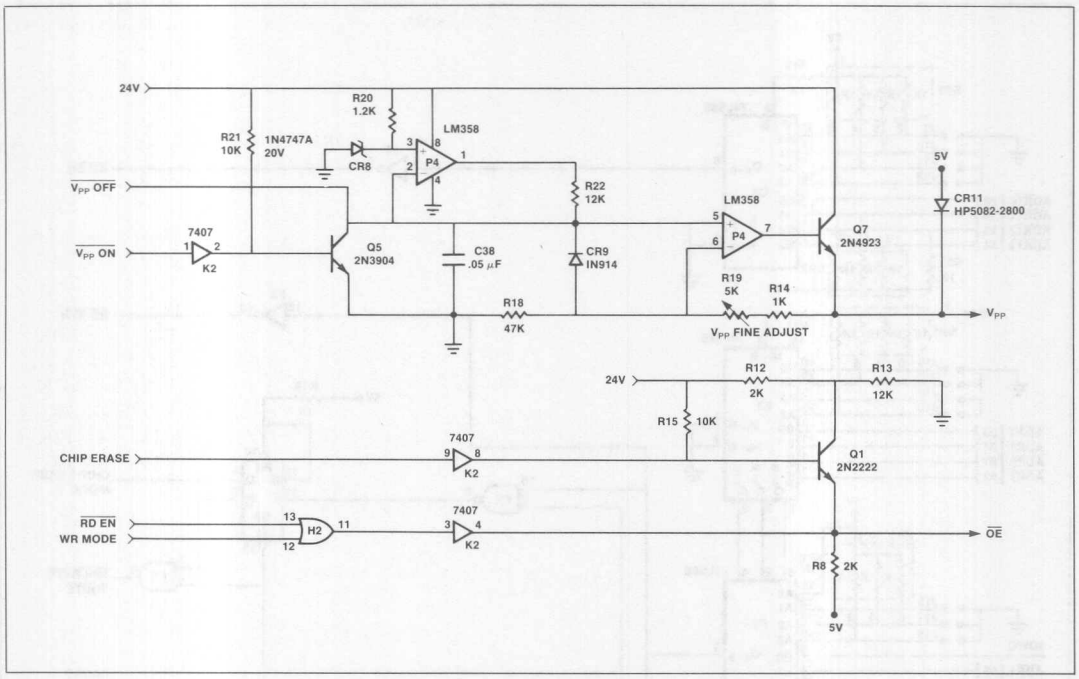


Figure 17. V_{PP} and OE Drivers

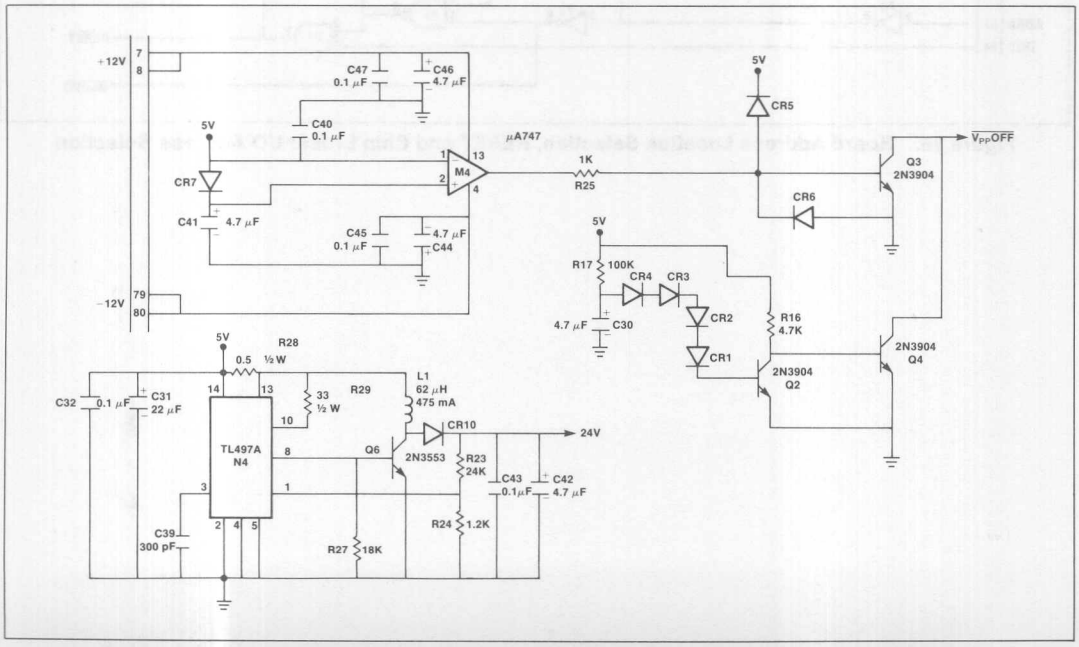


Figure 18. 5V to 23V Converter and Power Up/Power Down Write Protection Circuitry

ASSEMBLY INSTRUCTIONS

1. Install and solder the following Integrated Circuits:

A4	— 74S32	A2	— 74S04
B4	— 74S240	B2	— 74S00
D4	— 8283	C2	— 74S08
E4	— 8283	D2	— 74LS74
F4	— 8287	E2	— 74LS08
H4	— 8283	F2	— 74LS00
J4	— 8283	H2	— 74LS32
K4	— 8287	J2	— 74S30
A3	— 74S10	K2	— 7407
B3	— 74S05	A1	— 9602
C3	— 74LS85	B1	— PE-21216
D3	— 74LS85	C1	— 74LS393
E3	— 74LS85	D1	— 74LS74
F3	— 74LS04	E1	— 74LS74
J3	— 8286	F1	— 9602
		H1	— 9602
		L1	— 8282
		N4	— TL497A
		M4	— UA747
		P4	— LM358

2. Install and solder a 24-pin socket at K1.

3. Install and solder 28-pin sockets at M1, L2, N1, N2, P1, P2, R1 and R2.

4. Install and solder jumper pin pairs at the following locations:

J1	J5	J13	J16
J2	J6	J14	J17
J3	J7	J15	J18
J4	J8		J19
J9		J20	
J10			
J11			
J12			

5. Break in half 4 jumper pairs.

   Install and solder one jumper pair at J21. Install and solder one of the single jumper pins at J22.

   Install and solder another single jumper pin and one jumper pair at J23 and J24.

   Install and solder jumper pairs at JW7, JW3, and JW2.

   Install and solder single jumper pins at JW8, JW4, JW1, JW6, and JW5.

6. Install and solder the Dipswitch at location J1.

7. Install and solder resistors and resistor networks at the following locations:

H3	— 898-1-R1K	R16	— 4.7K
K3	— 898-1-R1K	R17	— 100K
RP1	— 784-1-R1K	R29	— 33, 1/2 W
RP2	— 784-1-R1K	R28	— 0.5, 1/2 W
RP3	— 784-1-R1K	R27	— 18K
RP4	— 784-1-R1K	R24	— 1.2K
RP5	— 784-1-R1K	R23	— 24K
R6	— 1K	R15	— 10K
R5	— 24K	R22	— 12K
R4	— 6.2K	R21	— 10K
R3	— 10K	R20	— 1.2K
R2	— 24K	R18	— 47K
R1	— 1K	R12	— 2K
R32	— 1K	R13	— 12K
R10	— 1K	R14	— 1K
R9	— 1K	R25	— 1K
R8	— 2K		
R19	— 5K Mini-potentiometer		
R31	— 1K		
R30	— 1K		

8. Install and solder capacitors in the following locations:

C13	— 0.1 μf	C19	— 0.1 μf
C12	— 0.1 μf	C18	— 0.1 μf
C11	— 0.1 μf	C17	— 0.1 μf
C10	— 0.1 μf	C16	— 0.1 μf
C9	— 0.33 μf	C15	— 0.1 μf
C8	— 4.7 μf	C28	— 0.1 μf
C7	— 20 pf	C27	— 0.1 μf
C6	— 0.1 μf	C26	— 0.1 μf
C5	— 0.1 μf	C25	— 0.1 μf
C4	— 0.1 μf	C24	— 0.1 μf
C3	— 0.1 μf	C37	— 0.1 μf
C2	— 0.1 μf	C49	— 22 μf
C1	— 0.1 μf	C36	— 0.1 μf
C23	— 0.1 μf	C35	— 0.1 μf
C22	— 0.1 μf	C34	— 0.1 μf
C21	— 0.1 μf	C33	— 0.1 μf
C20	— 0.1 μf	C48	— 22 μf

9. Install and solder the following diodes:

CR1	— 1N914	CR7	— 1N914
CR2	— 1N914	CR8	— 1N4747A
CR3	— 1N914	CR9	— 1N914
CR4	— 1N914	CR10	— 1N914
CR5	— 1N914	CR11	— HP5082-2800
CR6	— 1N914	CR12	— 1N914

10. Install and solder the following transistors:

Q1	—	2N2222A
Q2	—	2N3904
Q3	—	2N3904
Q4	—	2N3904
Q5	—	2N3904
Q6	—	2N3553

If hardware is provided or available, mount transistors in the following locations:

Q7	—	2N4923

Solder the leads of Q7 and Q8 to the solder pads on the board.

11. Install and solder the following capacitors:

C40	—	0.1 μf	C41	—	4.7 μf
C47	—	0.1 μf	C44	—	4.7 μf
C45	—	0.1 μf	C42	—	4.7 μf
C39	—	300 pf	C31	—	22 μf
C43	—	0.1 μf	C30	—	4.7 μf
C38	—	0.05 μf	C46	—	4.7 μf
C32	—	0.1 μf			

12. Install and solder the 62 μh RF choke at location L1 (just above the 60-pin edge connector).

APPENDIX A
JUMPER LIST

J1 $\overline{\text{INT0}}$
J2 $\overline{\text{INT1}}$
J3 $\overline{\text{INT2}}$
J4 $\overline{\text{INT3}}$
J5 $\overline{\text{INT4}}$
J6 $\overline{\text{INT5}}$
J7 $\overline{\text{INT6}}$
J8 $\overline{\text{INT7}}$

J9 (ADR10)
J10 (ADR11)
J11 (ADR12)
J12 (ADR13)

4 bit selection of one of 16 64K pages for board address
—Note: these ADRs are in HEX

J13 (ADR1)
J14 (ADR2)
J15 (ADR3)
J16 (ADR4)
J17 (ADR5)
J18 (ADR6)
J19 (ADR7)

Select I/O address for Chip Erase Mode (ADR 0=1) and RESET function (ADR 0=0)

J20 Jumpered for 16-bit wide data bus, Open for 8-bit wide data bus.

NOTE: The proper decoding algorithm for the data bus must be used in the BIPOLAR PROM decoder—refer to the PROM Array Address Configuration subsection of the Installation Instructions.

J21 (Select 4K/8K)
J22 (Select 2K)

MOS PROMs 1–4

J23 (Select 4K/8K)
J24 (Select 2K)

MOS PROMs 5–8

JW1
JW2
JW3
JW4
JW5
JW6
JW7
JW8

8-bit wide static RAM

APPENDIX B
BIP DECODER DATA FORMAT
3628A, 1K X 8, 000-3FFH

Data = all 1's at all locations not shown.
0 = Switch is on
1 = Switch is off

SW4	SW3	SW2	SW1	(Hex) Address	(Hex) Data	Decoding for
0	0	0	0	2	FE	2K X 8
				3	FD	
				6	FB	(2816, 2815 OR 2716)
				7	F7	
				A	EF	
				B	DF	
				E	BF	
				F	7F	8-BIT DATA BUS
0	0	0	1	42	FE	
				43	FD	4K X 8
				46	FE	
				47	FD	(2732, 2732A)
				4A	FB	
				4B	F7	
				4E	FB	
				4F	F7	8-BIT DATA BUS
				52	EF	
				53	DF	
				56	EF	
				57	DF	
				5A	BF	
				5B	7F	
				5E	BF	
				5F	7F	
0	0	1	0	82	FE	
				83	FD	
				86	FE	
				87	FD	8K X 8
				8A	FE	
				8B	FD	(2764)
				8E	FE	
				8F	FD	
				92	FB	
				93	F7	8-BIT DATA BUS
				96	FB	
				97	F7	
				9A	FB	
				9B	F7	
				9E	FB	
				9F	F7	
				A2	EF	
				A3	DF	
				A6	EF	
				A7	DF	
				AA	EF	
				AB	DF	
				AE	EF	
				AF	DF	
				B2	BF	
				B3	7F	

SW4	SW3	SW2	SW1	(Hex) Address	(Hex) Data	Decoding for
0	0	1	0	B6	BF	8K × 8
				B7	7F	8-BIT DATA BUS
				BA	BF	(Continued)
				BB	7F	
				BE	BF	
				BF	7F	
0	0	1	1	C0	FC	2K X 8
				C1	FD	16-BIT DATA BUS
				C2	FE	
				C4	F3	(2816, 2815, OR 2716)
				C5	F7	
				C6	FB	
				C8	CF	
				C9	DF	
				CA	EF	
				CC	3F	
				CD	7F	
				CE	BF	
0	1	0	0	100	FC	4K X 8
				101	FD	16-BIT DATA BUS
				102	FE	
				104	FC	(2732 OR 2732A)
				105	FD	
				106	FE	
				108	F3	
				109	F7	
				10A	FB	
				10C	F3	
				10D	F7	
				10E	FB	
				110	CF	
				111	DF	
				112	EF	
				114	CE	
				115	DF	
				116	EF	
				118	3F	
				119	7F	
				11A	BF	
				11C	3F	
				11D	7F	
				11E	BF	
0	1	0	1	140	FC	8K X 8
				141	FD	16-BIT DATA BUS
				142	FE	
				144	FC	
				145	FD	
				146	FE	
				148	FC	(2764)
				149	FD	
				14A	FE	
				14C	FC	
				14D	FD	
				14E	FE	
				150	F3	
				151	F7	
				152	FB	
				154	F3	
				155	F7	
				156	FB	

SW4	SW3	SW2	SW1	(Hex) Address	(Hex) Data	Decoding for
0	1	0	1	158	F3	8K × 8
				159	F7	16-BIT DATA BUS
				15A	FB	(Continued)
				15C	F3	
				15D	F7	
				15E	FB	
				160	CF	
				161	DF	
				162	EF	
				164	CF	
				165	DF	
				166	EF	
				168	CF	
				169	DF	
				16A	EF	
				16C	CF	
				16D	DF	
				16E	EF	
				170	3F	
				171	7F	
				172	BF	
				174	3F	
				175	7F	
				176	BF	
				178	3F	
				179	7F	
				17A	BF	
				17C	3F	
				17D	7F	
				17E	BF	
0	1	1	0	1A2	FE	LOCATED AT 8000H
				1A3	FD	8-BIT DATA BUS
				1A6	FB	
				1A7	F7	(2716, 2816 OR 2815)
				1AA	EF	
				1AB	DF	
				1AE	BF	
				1A7	7F	
0	1	1	1	1E0	FC	LOCATED AT 8000H
				1E1	FD	16-BIT DATA BUS
				1E2	FE	
				1E4	F3	(2716, 2816 OR 2815)
				1E5	F7	
				1E6	FB	
				1E8	CF	
				1E9	DF	
				1EA	EF	
				1EC	3F	
				1ED	7F	
				1EE	BF	

APPENDIX C
PARTS LIST

Table C-1. Integrated Circuits

Qty	Description
1	3628A-4
4	8283
1	8286
2	8287
3	74LS85
1	74LS393
3	9602
3	74LS74
1	74LS08
1	74S10
1	74S30
1	74LS32
1	74LS04
1	74LS00
1	74S00
1	74S05
1	8282
1	74S08
1	74S04
1	74S32
1	74S240
1	LM358
1	TTL Delay Line PE-21216
1	μA747
1	7407
1	TL497A
36	TOTAL

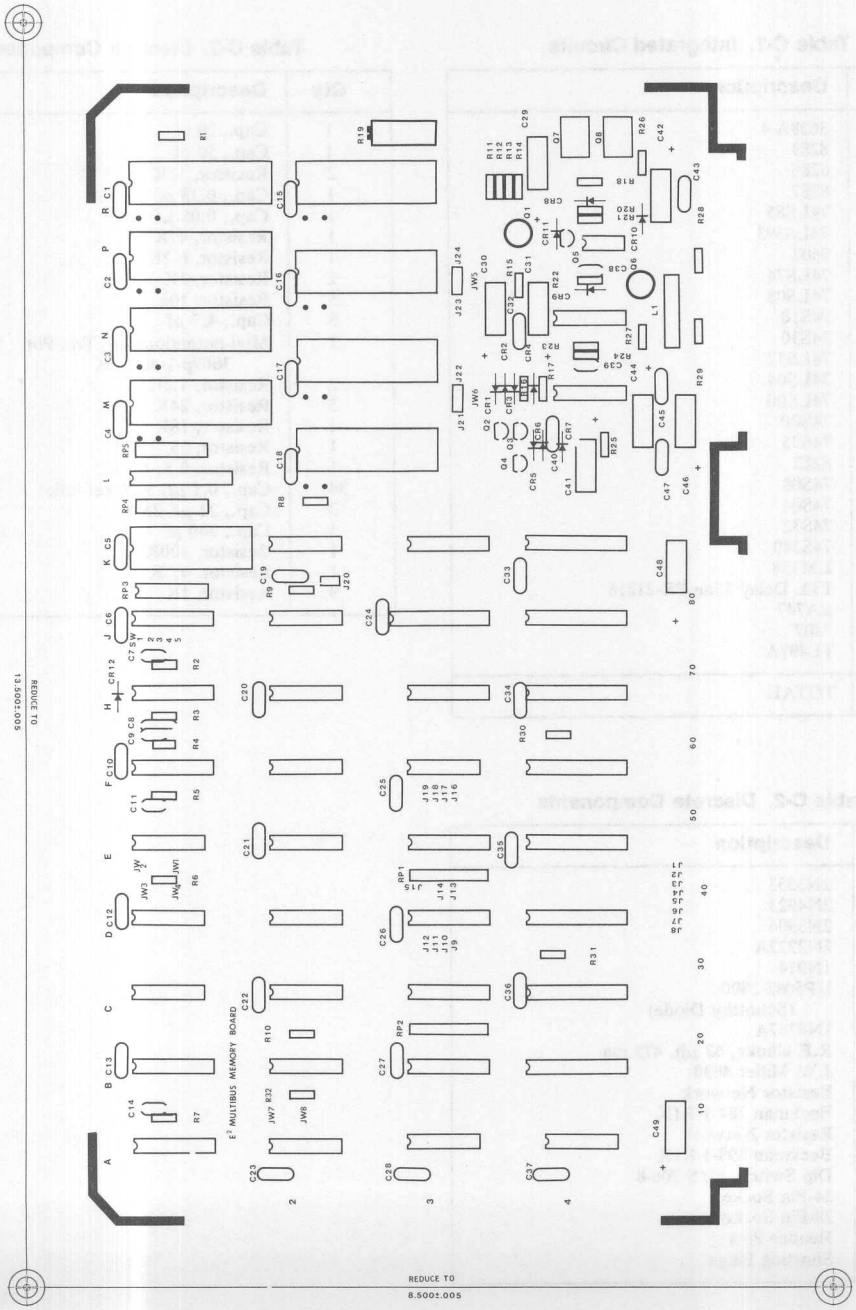
Table C-2. Discrete Components

Qty	Description
1	2N3553
1	2N4923
4	2N3904
1	2N2222A
11	1N914
1	HP5082-2800 (Schottky Diode)
1	1N4747A
1	R.F. Choke, 62 μh, 475 ma J.W. Miller 4630
5	Resistor Network Beckman 784-1-R1K
2	Resistor Network Beckman 898-1-R1K
1	Dip Switch, CTS 206-8
1	24-Pin Socket
8	28-Pin Sockets
29	Header Pins
18	Shorting Plugs

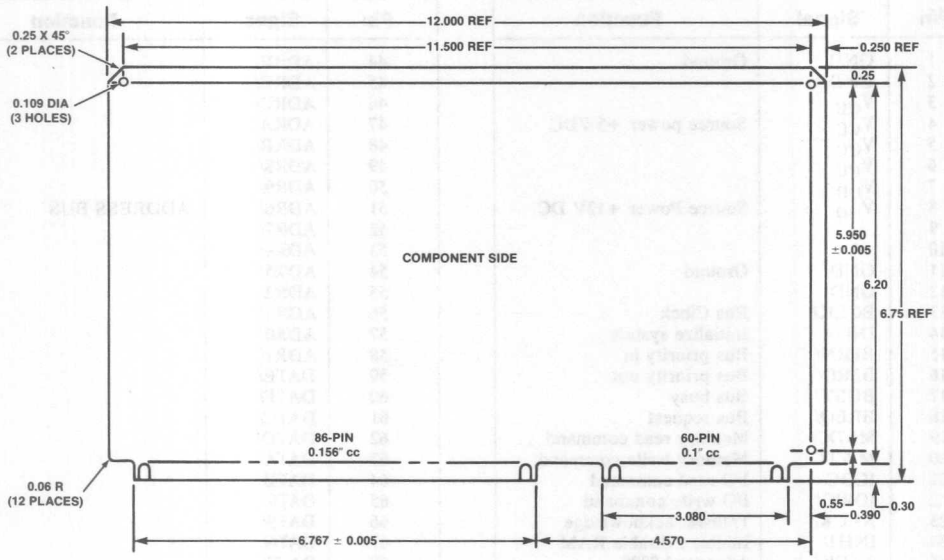
Table C-3. Discrete Components

Qty	Description
1	Cap., 20 pf
1	Cap., 50 pf
2	Resistor, 12K
1	Cap., 0.33 μf
1	Cap., 0.05 μf
1	Resistor, 47K
1	Resistor, 6.2K
2	Resistor, 2K
3	Resistor, 10K
6	Cap., 4.7 μf
1	Mini-potentiometer, TrimPot 3009p-1-502, 5K
2	Resistor, 1.2K
3	Resistor, 24K
1	Resistor, 18K
1	Resistor, 33, 1/2W
1	Resistor, 0.5, 1/2W
34	Cap., 0.1 μf, 50V, ceramic
3	Cap., 22 μf, 25V
1	Cap., 300 pf
1	Resistor, 100K
1	Resistor, 4.7K
9	Resistor, 1K

APPENDIX D
ASSEMBLY DRAWING



APPENDIX E
MULTIBUS PCB DIMENSIONS



AFN-02067A

APPENDIX F
MULTIBUS SIGNAL LIST

Pin	Signal	Function
1	GND	Ground
2	GND	
3	V_{CC}	
4	V_{CC}	Source power +5 VDC
5	V_{CC}	
6	V_{CC}	
7	V_{DD}	
8	V_{DD}	Source Power +12V DC
9		
10		
11	GND	Ground
12	GND	
13	BCLK/	Bus Clock
14	INIT/	Initialize system
15	BPRN/	Bus priority in
16	BPRO/	Bus priority out
17	BUSY/	Bus busy
18	BREQ/	Bus request
19	MRDC/	Memory read command
20	MWTC/	Memory write command
21	IORC/	I/O read command
22	IOWC/	I/O write command
23	XACK/	Transfer acknowledge
24	INH1/	Inhibit 1 disable RAM
25	AACK/	Advanced 8080 acknowledge
26		
27	BHEN/	Byte High Enable
28	AD10/	
29		
30	AD11/	
31		
32	AD12/	
33		
34	AD13/	
35	INT6/	
36	INT7/	
37	INT4/	External Interrupt
38	INT5/	Level Requests
39	INT2/	
40	INT3/	
41	INT0/	
42	INT1/	
43	ADRE/	ADDRESS BUS

Pin	Signal	Function
44	ADRF/	
45	ADRC/	
46	ADRD/	
47	ADRA/	
48	ADRB/	
49	ADR8/	
50	ADR9/	
51	ADR6/	ADDRESS BUS
52	ADR7/	
53	ADR4/	
54	ADR5/	
55	ADR2/	
56	ADR3/	
57	ADR0/	
58	ADR1/	
59	DATE/	
60	DATF/	
61	DATC/	
62	DATD/	
63	DATA/	
64	DATB/	
65	DAT8/	
66	DAT9/	
67	DAT6/	
68	DAT7/	
69	DAT4/	
70	DAT5/	
71	DAT2/	DATA BUS
72	DAT3/	
73	DAT0/	
74	DAT1/	
75	GND	
76	GND	Ground
77	V_{BB}	
78	V_{BB}	Source power −12 VDC
79		
80		
81	V_{CC}	
82	V_{CC}	Source power +5V
83	V_{CC}	
84	V_{CC}	
85	GND	
86	GND	Ground

APPENDIX G
BLANK DECODER CHARTS

SYSTEM ADDRESS A0-15 HEX	DECODER CIRCUIT INPUTS						CE'S								
	BAF	BAE	BAD	BAC	$\overline{BHEN}$	LA0	7	6	5	4	3	2	1	0	BYTE
0 X X X	0	0	0	0	1 1	0 1									L H
1 X X X	0	0	0	1	1 1	0 1									L H
2 X X X	0	0	1	0	1 1	0 1									L H
3 X X X	0	0	1	1	1 1	0 1									L H
4 X X X	0	1	0	0	1 1	0 1									L H
5 X X X	0	1	0	1	1 1	0 1									L H
6 X X X	0	1	1	0	1 1	0 1									L H
7 X X X	0	1	1	1	1 1	0 1									L H
8 X X X	1	0	0	0	1 1	0 1									L H
9 X X X	1	0	0	1	1 1	0 1									L H
A X X X	1	0	1	0	1 1	0 1									L H
B X X X	1	0	1	1	1 1	0 1									L H
C X X X	1	1	0	0	1 1	0 1									L H
D X X X	1	1	0	1	1 1	0 1									L H
E X X X	1	1	1	0	1 1	0 1									L H
F X X X	1	1	1	1	1 1	0 1									L H
X=HEX DIGITS	A5	A4	A3	A2	A1	A0	0 7	0 6	0 5	0 4	0 3	0 2	0 1	0 0	
	ADDRESS INPUTS 3628A						OUTPUTS								

L = LOW BYTE
H = HIGH BYTE

0 = ENABLE
1 = DISABLE

0 = NO SHORTING PLUG
X = SHORTING PLUG INSTALLED

8 BIT DATA BUS

LEAVE JUMPER J20 OPEN. (NO SHORTING PLUG)

INSTALL SHORTING PLUGS PER THE FOLLOWING TABLE:

DEVICE DENSITY = K BYTES

	JUMPERS			
	PROMS: 1-4		5-8	
	J21	J22	J23	J24
2K	0	X	0	X
4K/8K	X	0	X	0

SYSTEM ADDRESS A0–15 HEX	DECODER CIRCUIT INPUTS						$\overline{CE}$'S								
	BAF	BAE	BAD	BAC	$\overline{BHEN}$	LA0	7	6	5	4	3	2	1	0	BYTE
0XXX	0	0	0	0	0 1 0	0 0 1									W L H
1XXX	0	0	0	1	0 1 0	0 0 1									W L H
2XXX	0	0	1	0	0 1 0	0 0 1									W L H
3XXX	0	0	1	1	0 1 0	0 0 1									W L H
4XXX	0	1	0	0	0 1 0	0 0 1									W L H
5XXX	0	1	0	1	0 1 0	0 0 1									W L H
6XXX	0	1	1	0	0 1 0	0 0 1									W L H
7XXX	0	1	1	1	0 1 0	0 0 1									W L H
8XXX	1	0	0	0	0 1 0	0 0 1									W L H
9XXX	1	0	0	1	0 1 0	0 0 1									W L H
AXXX	1	0	1	0	0 1 0	0 0 1									W L H
BXXX	1	0	1	1	0 1 0	0 0 1									W L H
CXXX	1	1	0	0	0 1 0	0 0 1									W L H
DXXX	1	1	0	1	0 1 0	0 0 1									W L H
EXXX	1	1	1	0	0 1 0	0 0 1									W L H
FXXX	1	1	1	1	0 1 0	0 0 1									W L H
	A5	A4	A3	A2	A1	A0	0 7	0 6	0 5	0 4	0 3	0 2	0 1	0 0	
X = HEX DIGITS	ADDRESS INPUTS 3628A						OUTPUTS								

L = LOW BYTE
H = HIGH BYTE

0 = ENABLE
1 = DISABLE

0 = NO SHORTING PLUG
X = SHORTING PLUG INSTALLED

16 BIT DATA BUS

INSTALL SHORTING PLUG AT JUMPER J20.

INSTALL SHORTING PLUGS PER THE FOLLOWING TABLE:

DEVICE DENSITY = __K BYTES

	JUMPERS			
	PROMS: 1-4		5-8	
	J21	J22	J23	J24
2K	0	X	0	X
4K/8K	X	0	X	0

APPENDIX H
TEST DECODING ALGORITHMS FOR 2K X 8 MOS PROMs AT 8000H

SYSTEM ADDRESS A0-15 HEX	DECODER CIRCUIT INPUTS						$\overline{CE}$'S								
	BAF	BAE	BAD	BAC	$\overline{BHEN}$	LA0	7	6	5	4	3	2	1	0	BYTE
0 X X X	0	0	0	0	1	0									L
					1	1									H
1 X X X	0	0	0	1	1	0									L
					1	1									H
2 X X X	0	0	1	0	1	0									L
					1	1									H
3 X X X	0	0	1	1	1	0									L
					1	1									H
4 X X X	0	1	0	0	1	0									L
					1	1									H
5 X X X	0	1	0	1	1	0									L
					1	1									H
6 X X X	0	1	1	0	1	0									L
					1	1									H
7 X X X	0	1	1	1	1	0									L
					1	1									H
8 X X X	1	0	0	0	1	0								0	L
					1	1							0		H
9 X X X	1	0	0	1	1	0						0			L
					1	1					0				H
A X X X	1	0	1	0	1	0				0					L
					1	1			0						H
B X X X	1	0	1	1	1	0		0							L
					1	1	0								H
C X X X	1	1	0	0	1	0									L
					1	1									H
D X X X	1	1	0	1	1	0									L
					1	1									H
E X X X	1	1	1	0	1	0									L
					1	1									H
F X X X	1	1	1	1	1	0									L
					1	1									H
X=HEX DIGITS	A5	A4	A3	A2	A1	A0	0 7	0 6	0 5	0 4	0 3	0 2	0 1	0 0	
	ADDRESS INPUTS 3628A						OUTPUTS								

L = LOW BYTE
H = HIGH BYTE

0 = ENABLE
1 = DISABLE

0 = NO SHORTING PLUG
X = SHORTING PLUG INSTALLED

8 BIT DATA BUS

LEAVE JUMPER J20 OPEN. (NO SHORTING PLUG)

INSTALL SHORTING PLUGS PER THE FOLLOWING TABLE:

DEVICE DENSITY = 2K BYTES

	JUMPERS			
PROMS:	1-4		5-8	
	J21	J22	J23	J24
2K	0	X	0	X
4K/8K	X	0	X	0

SYSTEM ADDRESS A0–15 HEX	DECODER CIRCUIT INPUTS						$\overline{CE}$'S								BYTE
	BAF	BAE	BAD	BAC	$\overline{BHEN}$	LA0	7	6	5	4	3	2	1	0	
0XXX	0	0	0	0	0 1 0	0 0 1									W L H
1XXX	0	0	0	1	0 1 0	0 0 1									W L H
2XXX	0	0	1	0	0 1 0	0 0 1									W L H
3XXX	0	0	1	1	0 1 0	0 0 1									W L H
4XXX	0	1	0	0	0 1 0	0 0 1									W L H
5XXX	0	1	0	1	0 1 0	0 0 1									W L H
6XXX	0	1	1	0	0 1 0	0 0 1									W L H
7XXX	0	1	1	1	0 1 0	0 0 1									W L H
8XXX	1	0	0	0	0 1 0	0 0 1							0 0	0 0	W L H
9XXX	1	0	0	1	0 1 0	0 0 1					0 0	0 0			W L H
AXXX	1	0	1	0	0 1 0	0 0 1			0 0	0 0					W L H
BXXX	1	0	1	1	0 1 0	0 0 1	0 0	0 0							W L H
CXXX	1	1	0	0	0 1 0	0 0 1									W L H
DXXX	1	1	0	1	0 1 0	0 0 1									W L H
EXXX	1	1	1	0	0 1 0	0 0 1									W L H
FXXX	1	1	1	1	0 1 0	0 0 1									W L H
	A5	A4	A3	A2	A1	A0	0 7	0 6	0 5	0 4	0 3	0 2	0 1	0 0	
X = HEX DIGITS	ADDRESS INPUTS 3628A						OUTPUTS								

L = LOW BYTE
H = HIGH BYTE

0 = ENABLE
1 = DISABLE

0 = NO SHORTING PLUG
X = SHORTING PLUG INSTALLED

16 BIT DATA BUS

INSTALL SHORTING PLUG AT JUMPER J20.

INSTALL SHORTING PLUGS PER THE FOLLOWING TABLE:

DEVICE DENSITY 2K BYTES

	JUMPERS			
	PROMS: 1-4		PROMS: 5-8	
	J21	J22	J23	J24
2K	0	X	0	X
4K/8K	X	0	X	0

intel

APPLICATION NOTE

AP-137

September 1981

8298 Functional Specification and Firmware Description

Kevin Corbett and Randy Battat
Special Products Division
Applications Engineering

©Intel Corporation, 1981.

- **Supervises Writing, Erasing and Reading of Intel E^2PROMs**
- **Supports Up to 16K Bytes of E^2 Memory**
- **Provides Optimized Read/Write Commands for Efficient Data Transfers**
- **Microprocessor Peripheral, Compatible with iAPX 86/88, MCS®-80/85, MCS®-48 Families**
- **Provides Data Check to Minimize Necessary E^2 Writes**
- **Supports Interrupt or Polled Operation**

The 8298 Intelligent E^2PROM (Electrically Erasable Programmable Read Only Memory) Controller is a microprocessor peripheral designed to efficiently oversee reading, writing, and erasing of Intel E^2 devices. Up to 16K bytes of E^2PROM memory are supported by the 8298 with a few external components. These components include an 8243 I/O expander which provides address and control signals, a V_{PP} Switch to provide program pulses, and an $\overline{OE}$ switch to provide for erasure of a complete chip. The 8298 also provides local bus request and acknowledge signals enabling the E^2PROMs to couple directly with system busses.

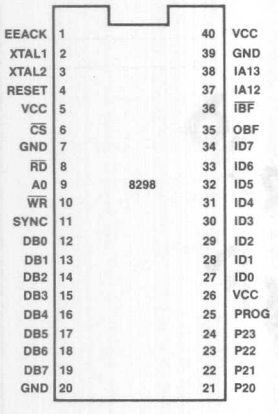


Figure 1. Pinouts of 8298

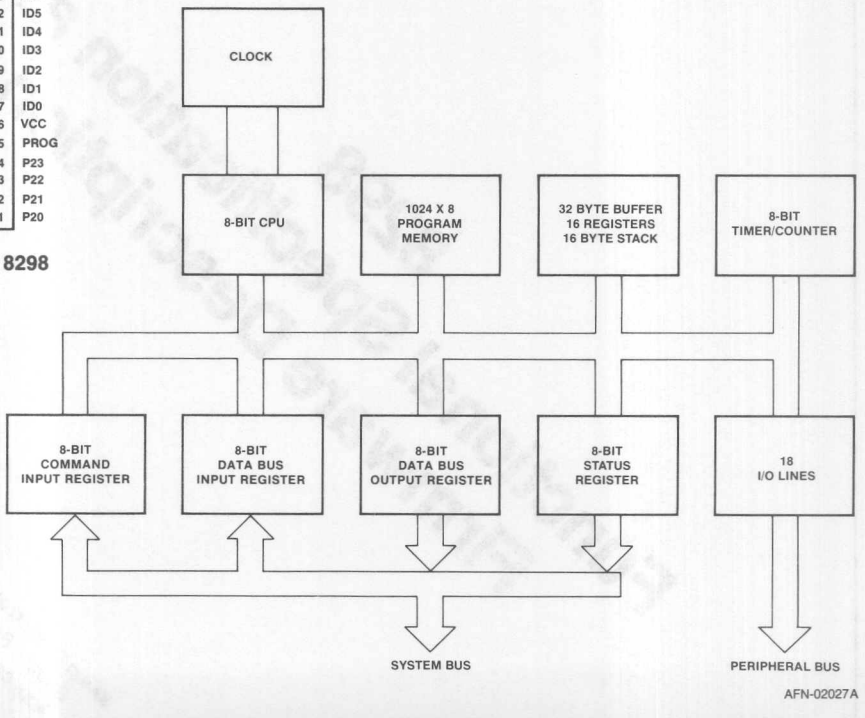


Figure 2. 8298 Block Diagram

Table 1. Pin Description (8298)

Symbol	Pin No.	Type	Function
EEACK	1	I	E^2 Acknowledge—a logical 1 on this pin indicates that the 8298 has access to the E^2PROM memory (via local bus).
XTAL1, XTAL2	2, 3	I	Inputs for a crystal, LC circuit, or external timing signal to determine internal oscillator frequency.
RESET	4	I	Used to initialize the chip to a known state during power on.
$\overline{CS}$	6	I	Chip Select Input—used to select the 8298 for other devices on the common data bus.
$\overline{RD}$	8	I	I/O read input which allows the master CPU to read from the 8298.
A0	9	I	Address Line—used to select between data and status registers during read operations and to distinguish between data and commands written to the 8298 during write operations.
$\overline{WR}$	10	I	I/O or Memory Write input which allows the master CPU to write the 8298.
SYNC	11	O	8298 cycle synchronization signal.
DB0–DB7	12–19	I/O	8 bidirectional lines used for communications between the central processor and the 8298 registers.
GND	7, 20, 39	P.S.	Circuit ground potential.
P20–P23	21–24	I/O	I/O information passed to the 8243 I/O expander.
PROG	25	O	Latching signal for 8243 I/O expander.
ID0–UD7	27–34	I/O	Internal data bus, 8 bidirectional data lines common to the 8298 and all E^2 devices.
OBF	35	O	Output Buffer Full—used to interrupt the host CPU when the data output buffer of the 8298 is full.
$\overline{TBF}$	36	O	Input Buffer Full—used to interrupt the host CPU when the 8298 has emptied the input buffer.
IA12, IA13	27, 38	O	Internal address bits 12, 13.
V_{CC}	5, 40, 26	P.S.	+5V supply input ±10%.
IA0–IA11	1–5, 17–23	I/O	Internal address bus—12 bidirectional address lines common to the 8298 and all E^2 devices.
P20–P23	8–11	I/O	I/O information passed by 8298.
PROG	7	O	I/O information latching signal.
$\overline{V_{PP}}$ EN	13	O	V_{PP} control—activates programming voltage to V_{PP} switch when low.
$\overline{CRD}$	14	O	Controller Read—active low enables E^2PROM output buffers when 8298 performs an E^2 read operation.
$\overline{EEN}$	15	O	E^2 Enable—active low enables E^2PROM devices.
$\overline{EEREQ}$	16	O	E^2 Request—active low requests 8298 access to E^2 devices. 8298 will not take control of E^2PROMs (via EEN) until EEACK is brought high.
V_{CC}	24	P.S.	±5V power supply, ±10%.
GND	12	P.S.	System ground potential.

GENERAL DESCRIPTION

E^2 Operation

The 8298 receives commands and data from the host CPU to perform E^2 write, erase, and read operations. (See Figure 2 for a block diagram of the 8298.)

Before any E^2 operation is performed, however, the 8298 requests access to the local E^2 memory bus by bringing EEREQ low. If the E^2 memory bank is not being accessed by another processor, the external hardware will raise EEACK high. Systems in which the 8298 is the only device connected to E^2 memory may tie EEACK high so the 8298 is always granted access.

The 8298 performs a read operation by outputting the address of the selected location and bringing $\overline{EEN}$ (E^2 Enable) and $\overline{CRD}$ (Controller Read) low. Data is then read through the internal data bus I/O lines of the 8298 and all control signals are returned to their inactive state. Figure 3 shows a general system interface diagram for reference.

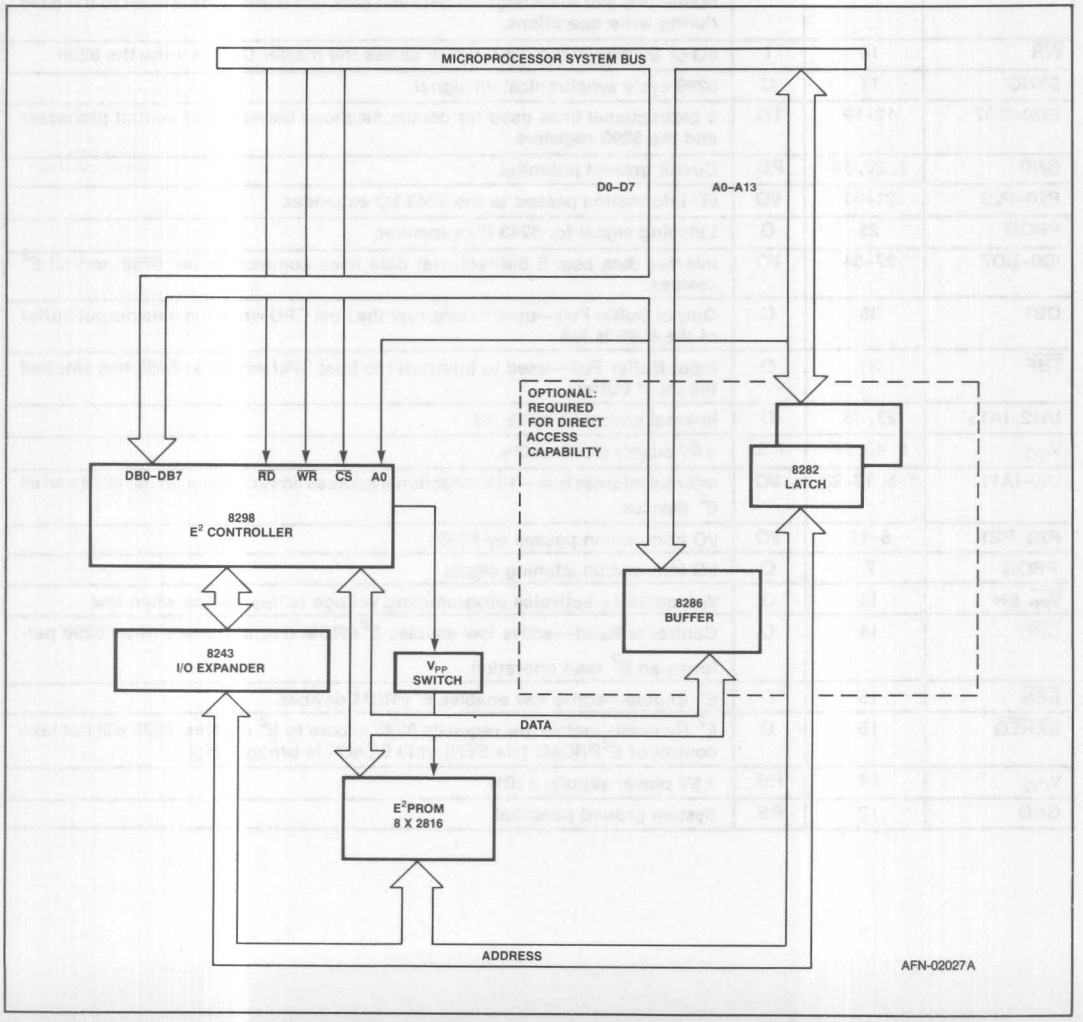


Figure 3. System Interfaces Diagram

Write and erase operations are performed in a similar manner. Address and data are output and $\overline{V_{PP}EN}$ is pulsed low for the duration of the write/erase cycle. V_{PP} pulse width is dependent upon clock oscillator frequency and is software programmable. The default value is 10 msec for a clock frequency of 6 MHz. At the end of the write/erase cycle, $\overline{V_{PP}EN}$ is brought inactive (high) and the 8298 delays while the 2816 programming voltage (V_{PP}) falls to 5V. This delay is guaranteed to be greater than 100 μsec at 6 MHz clock frequency; at lower frequencies the delay is longer.

When the 8298 receives a write command, the E^2 location to be written is first read. If the data is the same as the data currently in memory, the write operation is terminated. If not, the 8298 checks to see if a byte erase operation is necessary. An erase is necessary when the data to write contains a logical 1 where there is a logical 0 already stored. A byte erase operation is performed by writing all 1's to the selected byte. Following the optional erase cycle, the write cycle is performed.

HOST CPU INTERACTION

The host CPU and 8298 communicate with each other by means of four registers; two input and two output registers. Commands are issued when the host CPU writes a byte to the 8298 command register. Additional information required by the 8298 is transmitted by the host writing data bytes to the 8298 data input register. The 8298 transmits data back to the host (e.g., data read from E^2PROM) via its data output register. Data transfer is synchronized by interrupt lines or the status register which can be read by the host at any time. Register selection is done by the $\overline{RD}$ and $\overline{WR}$ signals and by A0 as shown in Figure 4.

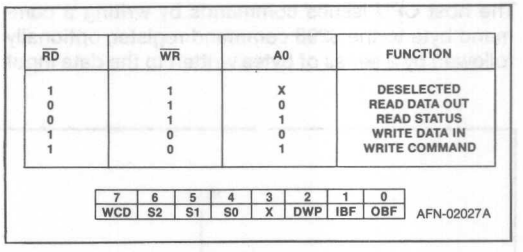


Figure 4. 8298 Register Selection

The status register can be read at any time to determine the state of the 8298. It is defined as follows:

OBF — Output Buffer Full—The data output register has data available for the host to read.

IBF — Input Buffer Full—The data input register or command register contains information not yet recognized by the 8298. The host CPU should never write to the 8298 when IBF = 0.

DWP — Direct Write Possible—The host CPU may perform a "Direct Write" if this bit is set and all other bits indicate that the UPI is waiting for a command.

WCD — Waiting for Command/Data—The 8298 is waiting for the host CPU to write a byte to its command or data input register.

S0, S1, S2 — Additional status information can be read at any time to determine the state of the 8298. It is defined as follows:

WCD	S2	S1	S0	OBF	Function
1	1	1	1	0	Waiting for command
0	0	0	0	0	Executing command
1	1	1	0	0	Illegal command—waiting for new command
0	0	X	X	1*	Illegal commmand transmitted during E^2 write cycle
1	0	X	X	1*	Command issued data expected
1	1	1	1	1	Read command complete—data output register contains data read
0	0	0	0	1	Series read command data available
0	1	0	1	0	Write cycle in progress

WCD	S2	S1	S0	OBF	Function
1	0	0	0	0	Waiting for data byte #1
1	0	0	1	0	Waiting for data byte #2
1	0	1	0	0	Waiting for data byte #3
1	9	1	1	0	Waiting for data byte #4

X = Don't care.
* = Data output register contains 10101010 to indicate illegal operation.

COMMANDS

The host CPU issues commands by writing a command byte to the 8298 command register, optionally followed by a series of bytes written to the data input register. The status register indicates the type of data expected and when the 8298 is ready for commands and data. Figure 5 shows a flowchart detailing the transmission of a command.

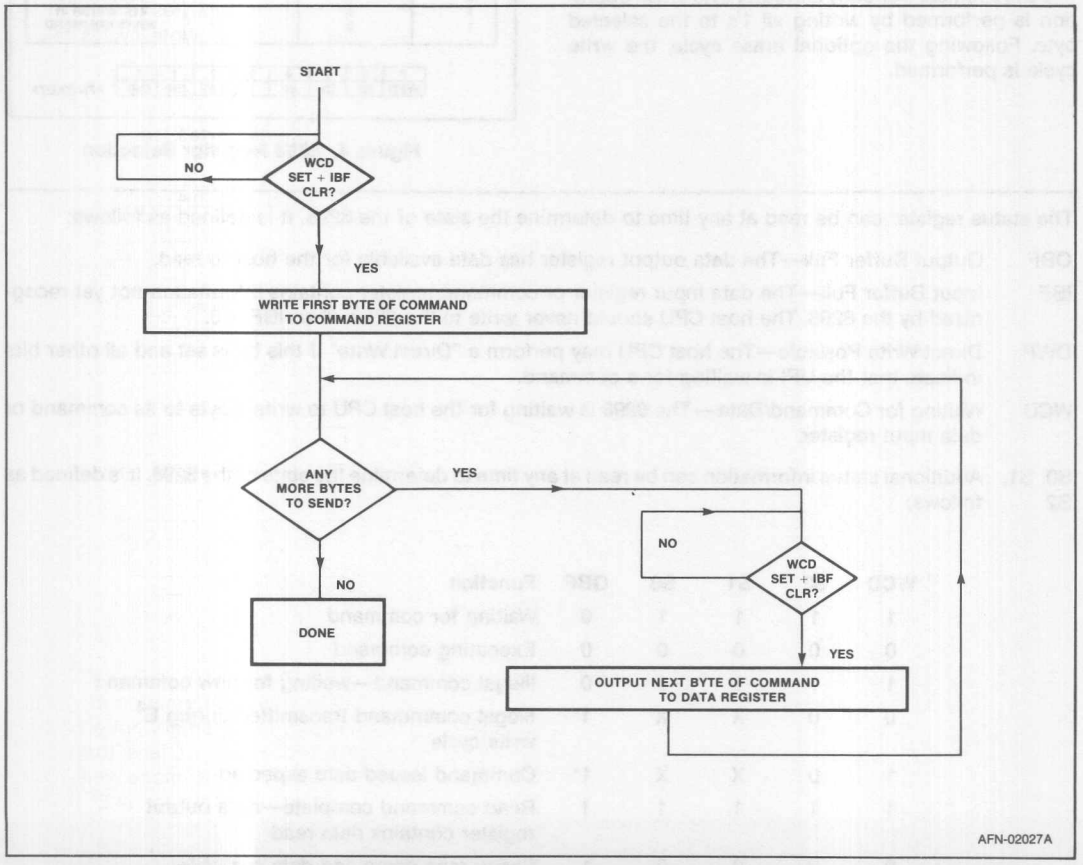


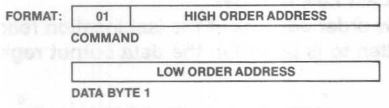
Figure 5. Flow Chart for Command Transmission

READ COMMANDS

The following commands are issued to read data from E^2 memory.

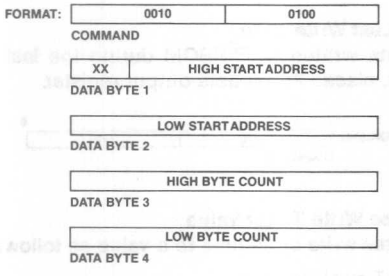
Indirect Read

Read the contents of a single location. Data read is passed to the host via the data output register. The command consists of the following sequence.



Series Read

Read up to 16,384 sequential locations. Data read is placed in the data output register. As soon as the host reads a byte from this register, it is reloaded with data read from the next location. Note that the host CPU should determine that data is available by verifying that the OBF bit is set in the status register.

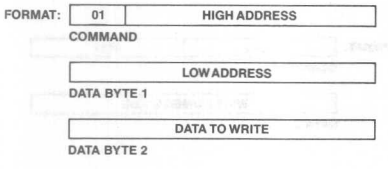


WRITE/ERASE COMMANDS

Five different write and erase commands are provided.

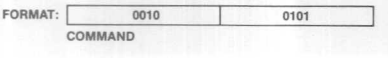
Indirect Write

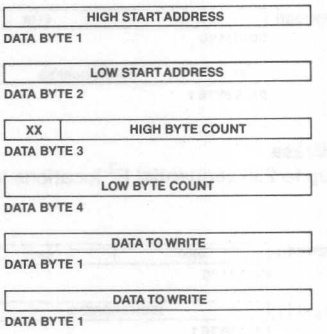
Write a single E^2 location



Series Write

Write up to 16,384 sequential locations with increasing addresses.

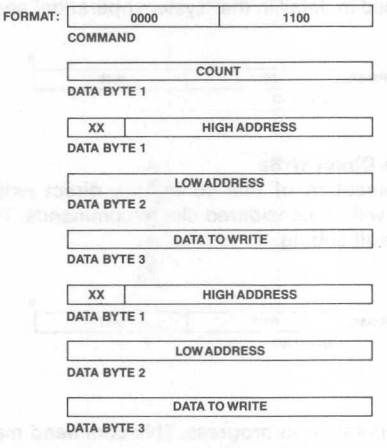




Note that the 8298 expects "data byte 1" for each byte of data transmitted. Also, the 8298 internally buffers up to 32 data bytes at a time. Therefore, series writes of less than 33 locations are performed efficiently because the host CPU outputs all bytes to write in a "burst" manner before actual E^2 write cycles take place.

Multiple Write

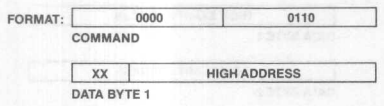
Write up to 256 non-sequential bytes.



Note the sequence (data byte 1, 2, 3, 1, 2, 3 . . .) of data in which the 8298 receives information. Up to 9 address/data combinations are internally buffered by the 8298. Thus multiple writes of less than 10 locations are performed in a "burst" mode where all bytes are buffered by the 8298 before actual E^2 write cycles take place.

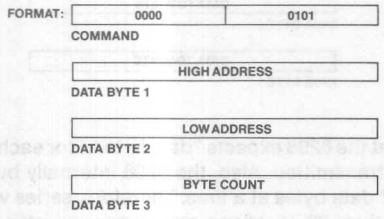
Chip Erase

Erase an entire E^2PROM. The high order address is required to select the device to be erased.



Block Erase

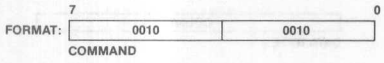
Erase up to 256 sequential E^2 locations in increasing order.



UTILITY COMMANDS

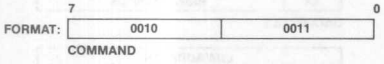
Enable Direct Write

Enables direct writes. A direct write is performed when the 8298 received data when it is waiting for a command and direct writes are enabled. This topic is discussed in detail in the "system operation" section.



Disable Direct Write

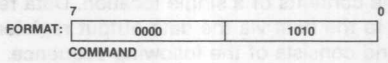
After execution of this command, direct write attempts will be considered illegal commands. This is the default setting.



Abort

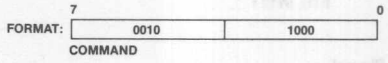
Abort operation in progress. This command may be given at any time. If the 8298 is in the middle of a write cycle, writing is stopped, V_{PP} is brought low, and after a delay while V_{PP} falls to 5V (approximately 100 μsec at 6 MHz), the 8298 waits for another command.

After issuing an abort command, the 8298 may go back and, using the commands described below, determine the address and data of the interrupted write operation.



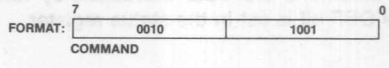
Read Last Low Address

The low order address of the last location read from or written to is placed in the data output register.



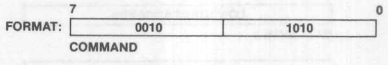
Read Last High Address

The high order address of the last location read from or written to is placed in the data output register.



Read Last Write Data

The data written to E^2PROM during the last write cycle is placed in the data output register.



Initialize Write Timer Value

Initializes write cycle time to a value as follows:

$$\text{Write Timer Value}_{10} = 256 - \frac{(\text{Clock Frequency (Hz)} * \text{Write Time (Sec)})}{480}$$

Thus, for a 10 Msec write time with a 6 MHz clock, the write timer value should be 131_{10}. This is the default value.

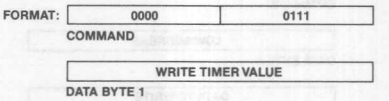


Table 2

Command	# of Bytes for Commands	Byte#	Format	Next Status
INDIRECT READ	2	1 2	01 High Address Low Address	DB1 OBF
SERIES READ	5	1 2 3 4 5	0010 0100 XX High Start Address Low Start Address High Byte Count Low Byte Count	DB1 DB2 DB3 DB4 OBF
INDIRECT WRITE	2+ WRITE DATA	1 2 3	01 High Address Low Address DATA TO WRITE	DB1 DB2 WCD
SERIES WRITE	5+ WRITE DATA	1 2 3 4 5	0010 0101 XX High Start Address Low Start High Byte Count Low Byte Count DATA TO WRITE	DB1 DB2 DB3 DB4 DB1 DB1/WCD NOTE 1
MULTIPLE WRITE	2+ ADDRESS/WRITE DATA	1 2 * * *	0000 1100 Byte Count XX High Address Low Address DATA TO WRITE	DB1 DB1 DB2 DB3 DB1/WCD NOTE 2
BLOCK ERASE	4	1 2 3 4	0000 0101 High Start Address Low Start Address Byte Count	DB1 DB2 DB3 WCD
CHIP ERASE	2	1 2	0000 0110 XX High Address	DB1 WCD NOTE 3
ENABLE DIRECT WRITE	1	1	0010 0010	WCD
DISABLE DIRECT WRITE	1	1	0010 0011	WCD
ABORT	1	1	0000 1010	WCD
READ LAST LOW ADDRESS	1	1	0010 1000	WCD
READ LAST HIGH ADDRESS	1	1	0010 1001	WCD
INITIALIZE WRITE TIMER VALUE	2	1 2	0000 0111 Write Timer Value	DB1 WCD

*ADDRESS + WRITE DATA

NOTES:

1. SERIES WRITE returns a status of DB1 whenever it is waiting for write data. When BYTE COUNT bytes have been sent, it will return a status of WCD. See SERIES WRITE command, under Indirect Configuration Commands, for more information.
2. MULTIPLE WRITE works in a loop format once the command and byte count are received. It requires groups of three bytes and cycles through the status as: -DB1—DB2—DB3- until "count" groups of bytes are received. It then returns a status of WCD.
3. High address is the high-order six bits of the highest address in an individual E^2PROM. An example might be: FOR 7FFH as the highest address in an E^2PROM, to erase would require sending 0000 0111 as the high address.

E^2PROM INTERFACE

The 8298 requires a few external components to drive E^2PROMs. These components are organized as functional blocks as follows:

V_{PP} Switch

Used to gate 21 V to V_{PP} line of E^2PROMs for writing and erasing. See AP 102.

OE Switching

Used to gate the +12V to E^2PROM OE line for the chip erase function. The OE switch is also responsible for pulling OE low during Read operation.

Direct Access Circuits

Required only if the 8298 is to support direct reading and writing of E^2PROM.

Direct reading is a function whereby the host CPU treats E^2 memory as EPROM memory and reads directly without the supervision of the 8298.

Direct writing is similar to direct reading in that the host CPU thinks it is writing to RAM. External circuitry forces a "write data register" operation to the 8298, so the data to write is latched in the 8298 data input register. External latches connected in parallel with 8298 Internal Address Lines latch the address. The 8298 reads the contents of these latches in order to determine the destination address of the write cycle. The E^2 controller seizes the internal busses, performs the write operation, and signals the host CPU when done.

The 8298 recognizes a direct write command when it is waiting for a command, *data* is received, and direct writing is enabled.

The advantage of the direct write command is that the host CPU can use its memory reference instructions to access E^2 memory.

Chip Erase Signal

The chip erase signal is used by the OE switch to gate +12V to the OE line of E^2PROMs during the chip erase command.

This signal is multiplexed with ID7 (internal data, bit 7) and should be latched on a high-to-low transition of EEN.

SYSTEM OPERATION

The 8298 can communicate with the rest of the system using the two mechanisms described below:

1. I/O Polling. The host CPU repeatedly reads the 8298 status register to determine when commands and data can be transmitted.

2. Interrupts: The 8298 WCD and OBF lines are used to interrupt the host processor when the 8298 is ready to accept input or has output available.

Reset and Power-Up Procedure

Reset is used to bring the 8298 into a known state upon power-up. It must be held low at least 10 msec after the power supply is within tolerance.

APPENDIX A: 8298 ELECTRICAL CHARACTERISTICS

ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature Under Bias . . . . . 0°C to 70°C
Storage Temperature . . . . . – 65°C to + 150°C
Voltage on Any Pin With Respect to Ground . . . . . 0.5V to + 7V
Power Dissipation . . . . . 1.5 Watt

*COMMENT: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

D.C. AND OPERATING CHARACTERISTICS

(T_A=0°C to 70°C, V_{SS}=0V: 8298; V_{CC}=V_{DD}=+5V±5%)

Symbol	Parameter	Min.	Max.	Unit	Test Conditions
V_{IL}	Input Low Voltage (Except XTAL1, XTAL2, $\overline{RESET}$)	– 0.5	0.8	V	
V_{IL1}	Input Low Voltage (XTAL1, XTAL2, $\overline{RESET}$)	– 0.5	0.6	V	
V_{IH}	Input High Voltage (Except XTAL1, XTAL2, $\overline{RESET}$)	2.2	V_{CC}		
V_{IH1}	Input High Voltage (XTAL1, XTAL2, $\overline{RESET}$)	3.8	V_{CC}	V	
V_{OL}	Output Low Voltage (D_0-D_7)		0.45	V	I_{OL} = 2.0 mA
V_{OL1}	Output Low Voltage ($P_{10}P_{17}$, $P_{20}P_{27}$, Sync)		0.45	V	I_{OL} = 1.6 mA
V_{OL2}	Output Low Voltage (Prog)		0.45	V	I_{OL} = 1.0 mA
V_{OH}	Output High Voltage (D_0-D_7)	2.4		V	I_{OH} = – 400 μA
V_{OH1}	Output High Voltage (All Other Outputs)	2.4		V	I_{OH} = – 50 μA
I_{IL}	Input Leakage Current (T_0, T_1, $\overline{RD}$, $\overline{WR}$, $\overline{CS}$, A_0, EA)		± 10	μA	$V_{SS} \le V_{IN} \le V_{CC}$
I_{OZ}	Output Leakage Current (D_0-D_7, High Z State)		± 10	μA	V_{SS} + 0.45 ≤ $V_{IN} \le V_{CC}$
I_{LI}	Low Input Load Current ($P_{10}P_{17}$, $P_{20}P_{27}$)		0.5	mA	V_{IL} = 0.8V
I_{LI1}	Low Input Load Current ($\overline{RESET}$, $\overline{SS}$)		0.2	mA	V_{IL} = 0.8V
I_{DD}	V_{DD} Supply Current		15	mA	Typical = 5 mA
I_{CC} + I_{DD}	Total Supply Current		125	mA	Typical = 60 mA

A.C. CHARACTERISTICS

(T_A=0°C to 70°C, V_{SS}=0V: 8298; V_{CC}=V_{DD}=+5V ±5%)

REGISTER READ

Symbol	Parameter	Min.	Max.	Unit	Test Conditions
t_{AR}	$\overline{CS}$, A_0 Setup to $\overline{RD}$↓	0		ns	
t_{RA}	$\overline{CS}$, A_0 Hold After $\overline{RD}$↑	0		ns	
t_{RR}	$\overline{RD}$ Pulse Width	250		ns	
t_{AD}	$\overline{CS}$, A_0 to Data Out Delay		225	ns	C_L = 150 pF
t_{RD}	$\overline{RD}$↓ to Data Out Delay		225	ns	C_L = 150 pF
t_{DF}	$\overline{RD}$↑ to Data Float Delay		100	ns	
t_{CY}	Cycle Time (Except 8298)	2.5	15	μs	6.0 MHz XTAL
t_{CY}	Cycle Time (8298)	4.17	15	μs	3.6 MHz XTAL

REGISTER WRITE

Symbol	Parameter	Min.	Max.	Unit	Test Conditions
t_{AW}	$\overline{CS}$, A_0 Setup to $\overline{WR}$↓	0		ns	
t_{WA}	$\overline{CS}$, A_0 Hold After $\overline{WR}$↑	0		ns	
t_{WW}	$\overline{WR}$ Pulse Width	250		ns	
t_{DW}	Data Setup to $\overline{WR}$↑	150		ns	
t_{WD}	Data Hold After $\overline{WR}$↑	0		ns	

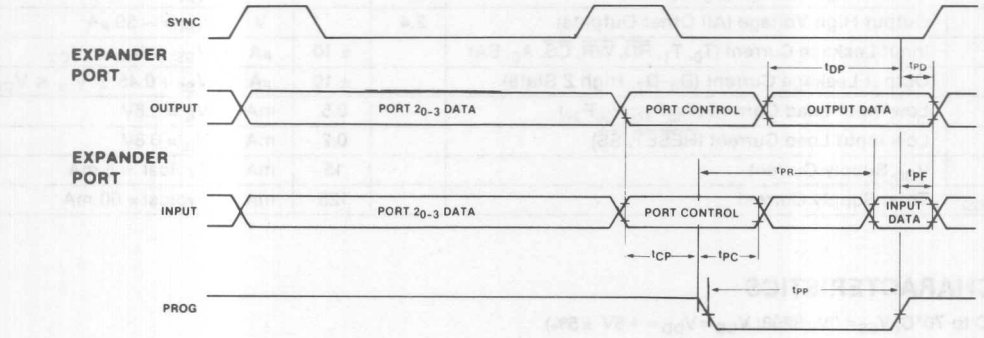
APPENDIX A: 8298 ELECTRICAL CHARACTERISTICS

A.C. CHARACTERISTICS—PORT 2

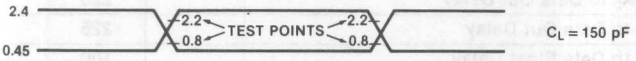
(T_A=0°C to 70°C: 8298; V_{CC}=+5V ±5%)

Symbol	Parameter	Min.	Max.	Unit	Test Conditions
t_{CP}	Port Control Setup Before Falling Edge of PROG	110		ns	
t_{PC}	Port Control Hold After Falling Edge of PROG	100		ns	
t_{PR}	PROG to Time P2 Input Must Be Valid		810	ns	
t_{PF}	Input Data Hold Time	0	150	ns	
t_{DP}	Output Data Setup Time	250		ns	
t_{PD}	Output Data Hold Time	65		ns	
t_{PP}	PROG Pulse Width	1200		ns	

PORT 2 TIMING



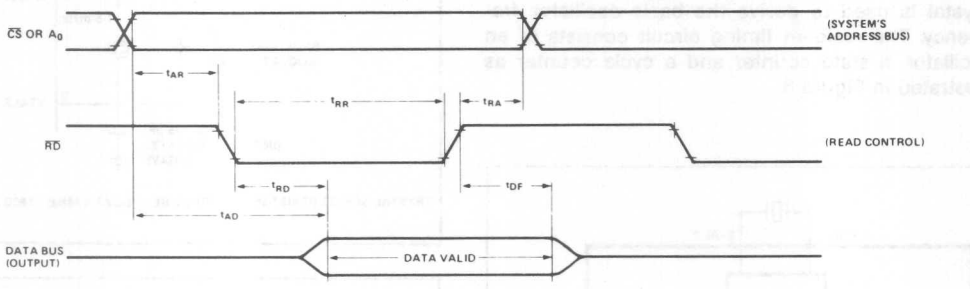
INPUT AND OUTPUT WAVEFORMS FOR A.C. TESTS



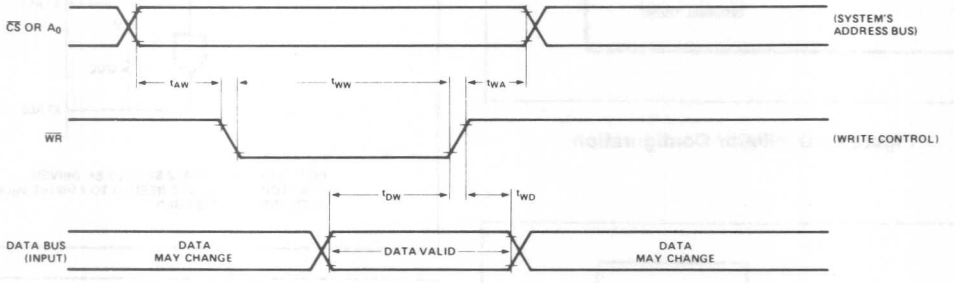
APPENDIX A: 8298 ELECTRICAL CHARACTERISTICS

WAVEFORMS

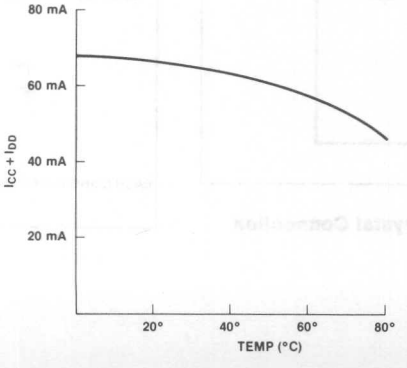
REGISTER READ OPERATION



REGISTER WRITE OPERATION



TYPICAL 8298 CURRENT



AFN-02027A

APPENDIX A: 8298 ELECTRICAL CHARACTERISTICS

OSCILLATOR AND TIMING CIRCUITS

The 8298's internal timing generation is controlled by a self-contained oscillator and timing circuit. A 6 MHz crystal is used to derive the basic oscillator frequency. The resident timing circuit consists of an oscillator, a state counter and a cycle counter as illustrated in Figure 6.

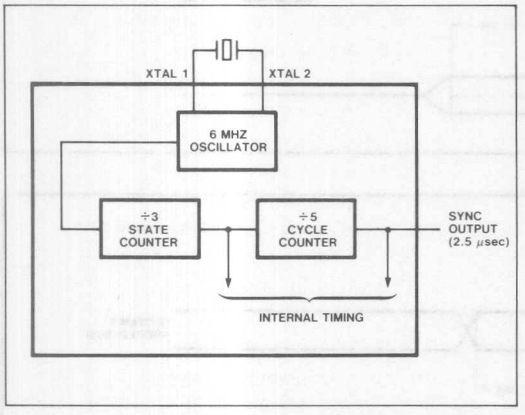


Figure 6. Oscillator Configuration

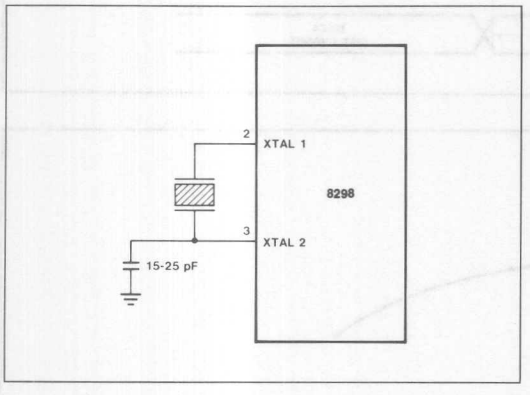
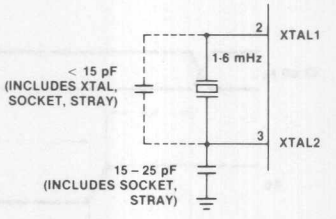


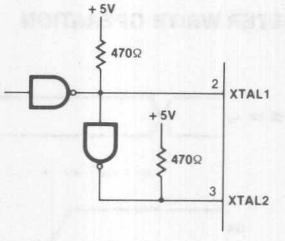
Figure 7. Recommended Crystal Connection

CRYSTAL OSCILLATOR MODE



CRYSTAL SERIES RESISTANCE SHOULD BE <75Ω AT 6 MHz; <180Ω AT 3.6 MHz.

DRIVING FROM EXTERNAL SOURCE

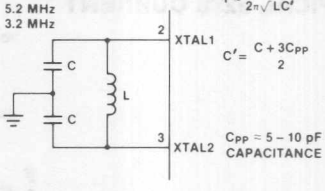


BOTH XTAL1 AND XTAL2 SHOULD BE DRIVEN. RESISTORS TO V_{CC} ARE NEEDED TO ENSURE $V_{IH} = 3.8V$ IF TTL CIRCUITRY IS USED.

LC OSCILLATOR MODE

L	C	NOMINAL f
45 μH	20 pF	5.2 MHz
120 μH	20 pF	3.2 MHz

$$f \approx \frac{1}{2\pi\sqrt{LC'}}$$



$$C' = \frac{C + 3C_{PP}}{2}$$

$C_{PP} \approx 5 - 10$ pF PIN-TO-PIN CAPACITANCE

EACH C SHOULD BE APPROXIMATELY 20 pF, INCLUDING STRAY CAPACITANCE

APPENDIX A: 8298 ELECTRICAL CHARACTERISTICS

PROGRAMMING, VERIFYING, AND ERASING THE 8298 EPROM

Programming Verification

In brief, the programming process consists of: activating the program mode, applying an address, latching the address, applying data, and applying a programming pulse. Each word is programmed completely before moving on to the next and is followed by a verification step. The following is a list of the pins used for programming and a description of their functions:

Pin	Function
XTAL 1	Clock Input (1 to 6MHz)
$\overline{\text{Reset}}$	Initialization and Address Latching
Test 0	Selection of Program or Verify Mode
EA	Activation of Program/Verify Modes
BUS	Address and Data Input Data Output During Verify
P20-1	Address Input
V_{DD}	Programming Power Supply
PROG	Program Pulse Input

WARNING:

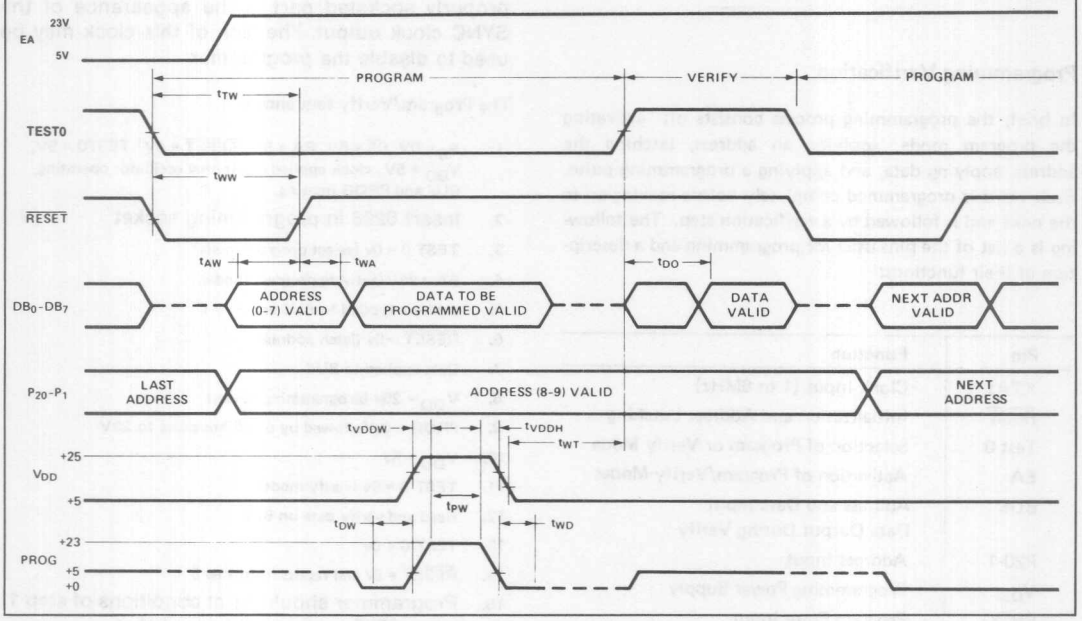
An attempt to program a missocketed 8298 will result in severe damage to the part. An indication of a properly socketed part is the appearance of the SYNC clock output. The lack of this clock may be used to disable the programmer.

The Program/Verify sequence is:

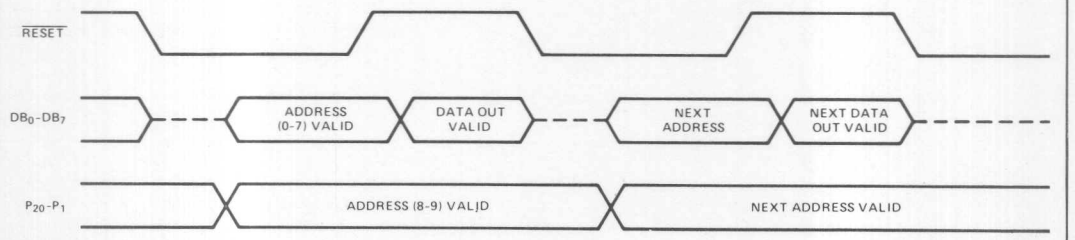
1. A_0 = 0V, $\overline{CS}$ = 5V, EA = 5V, $\overline{RESET}$ = 0V, TEST0 = 5V, V_{DD} = 5V, clock applied or internal oscillator operating, BUS and PROG floating.
2. Insert 8298 in programming socket
3. TEST 0 = 0v (select program mode)
4. EA = 23V (activate program mode)
5. Address applied to BUS and P20-1
6. $\overline{RESET}$ = 5v (latch address)
7. Data applied to BUS
8. V_{DD} = 25v (programming power)
9. PROG = 0v followed by one 50ms pulse to 23V
10. V_{DD} = 5v
11. TEST 0 = 5v (verify mode)
12. Read and verify data on BUS
13. TEST 0 = 0v
14. $\overline{RESET}$ = 0v and repeat from step 5
15. Programmer should be at conditions of step 1 when 8298 is removed from socket.

APPENDIX A: 8298 ELECTRICAL CHARACTERISTICS
WAVEFORMS FOR PROGRAMMING

COMBINATION PROGRAM/VERIFY MODE (EPROM'S ONLY)



VERIFY MODE (ROM/EPROM)



NOTES:
1. PROG MUST FLOAT IF EA IS LOW (i.e., =23V), OR IF T0 =5V FOR THE 8298.
2. XTAL1 AND XTAL2 DRIVEN BY 3.6 MHz CLOCK WILL GIVE 4.17 μsec t_{CY}. THIS IS ACCEPTABLE FOR 8298 PARTS AS WELL AS STANDARD PARTS.
3. AO MUST BE HELD LOW (i.e., =0V) DURING PROGRAM/VERIFY MODES.

The 8298 EPROM can be programmed by either of two Intel products:

1. PROMPT-48 Microcomputer Design Aid, or
2. Universal PROM Programmer (UPP series) peripheral of the Intellec® Development System with a UPP-848 Personality Card.

APPENDIX B: 8243 ELECTRICAL CHARACTERISTICS

ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature Under Bias . . . . . 0°C to 70°C
Storage Temperature . . . . . -65°C to +150°C
Voltage on Any Pin
With Respect to Ground . . . . . -0.5V to +7V
Power Dissipation . . . . . 1 Watt

COMMENT: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

D.C. AND OPERATING CHARACTERISTICS

(8243: T_A = 0°C to 70°C, V_{CC} = 5V ± 10%)

SYMBOL	PARAMETER	MIN.	TYP.	MAX.	UNITS	TEST CONDITIONS
V_{IL}	Input Low Voltage	-0.5		0.8	V	
V_{IH}	Input High Voltage	2.0		V_{CC}+0.5	V	
V_{OL1}	Output Low Voltage Ports 4-7			0.45	V	I_{OL} = 5 mA*
V_{OL2}	Output Low Voltage Port 7			1	V	I_{OL} = 20 mA
V_{OH1}	Output High Voltage Ports 4-7	2.4			V	I_{OH} = 240μA
I_{IL1}	Input Leakage Ports 4-7	-10		20	μA	V_{in} = V_{CC} to 0V
I_{IL2}	Input Leakage Port 2, CS, PROG	-10		10	μA	V_{in} = V_{CC} to 0V
V_{OL3}	Output Low Voltage Port 2			45	V	I_{OL} = 0.6 mA
I_{CC}	V_{CC} Supply Current		10	20	mA	
V_{OH2}	Output Voltage Port 2	2.4				I_{OH} = 100μA
I_{OL}	Sum of all IO_L from 16 Outputs			80	mA	5 mA Each Pin

*See following graph for additional sink current capability

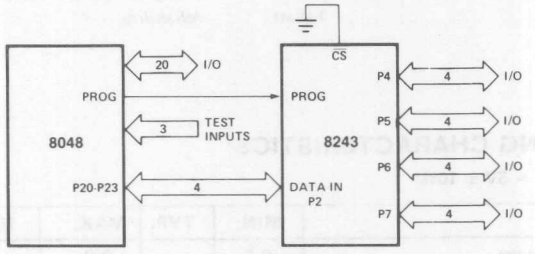
A.C. CHARACTERISTICS

(8243: T_A = 0°C to 70°C, V_{CC} = 5V ± 10%)

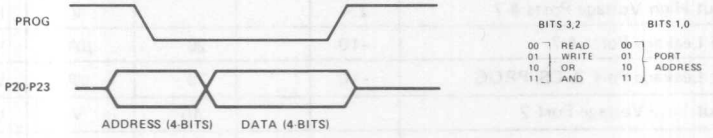
SYMBOL	PARAMETER	MIN.	MAX.	UNITS	TEST CONDITIONS
t_A	Code Valid Before PROG	100		ns	80 pF Load
t_B	Code Valid After PROG	60		ns	20 pF Load
t_C	Data Valid Before PROG	200		ns	80 pF Load
t_D	Data Valid After PROG	20		ns	20 pF Load
t_H	Floating After PROG	0	150	ns	20 pF Load
t_K	PROG Negative Pulse Width	700		ns	
t_{CS}	CS Valid Before/After PROG	50		ns	
t_{PO}	Ports 4-7 Valid After PROG		700	ns	100 pF Load
t_{LP1}	Ports 4-7 Valid Before/After PROG	100		ns	
t_{ACC}	Port 2 Valid After PROG		650	ns	80 pF Load

APPENDIX B: 8243 ELECTRICAL CHARACTERISTICS

EXPANDER INTERFACE

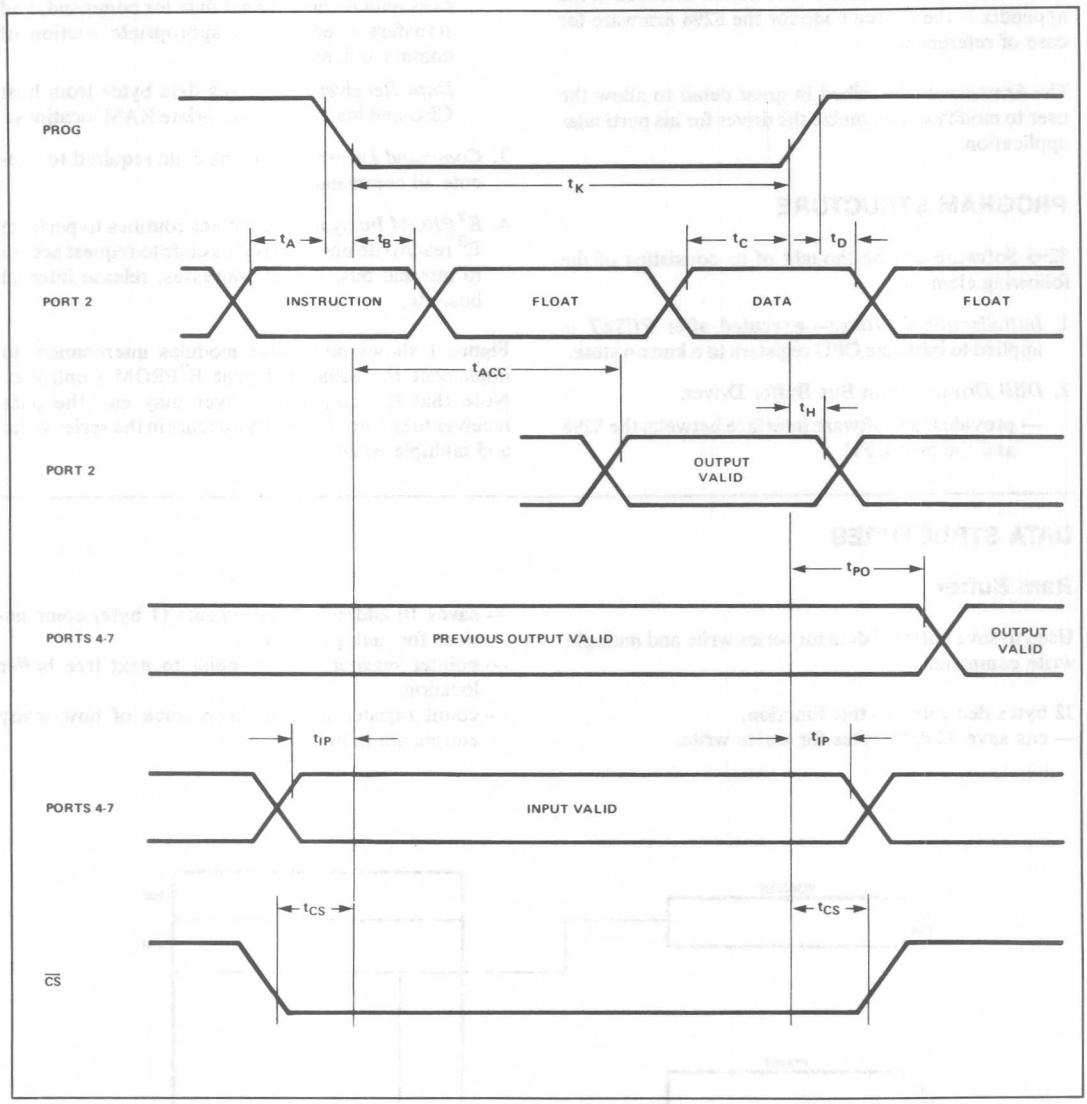


OUTPUT EXPANDER TIMING



APPENDIX B: 8243 ELECTRICAL CHARACTERISTICS

WAVEFORM



AFN-02027A

INTRODUCTION

The following is a description of the Firmware used in the 8298 E^2PROM Interface Controller. Included in the appendix is the source code for the 8298 firmware for ease of reference.

The firmware is described in great detail to allow the user to modify or customize the driver for his particular application.

PROGRAM STRUCTURE

8298 Software can be thought of as consisting of the following elements:

1. *Initialization Routines*—executed after *RESET* is applied to initialize CPU registers to a known state.
2. *DBB Driver*—Data Bus Buffer Driver.
   — provides the software interface between the 8298 and the host CPU.
   — handles receiving all commands and data.
   — DBB driver consists of two sections:

   Command Interpreter—Receives commands, calls data receiver to get data for command, and transfers execution to appropriate section of command driver.

   Data Receiver—Receives data bytes from host CPU and loads into appropriate RAM locations.
3. *Command Driver*—Contains code required to execute all commands.
4. *E^2PROM Interface*—Contains routines to perform E^2 read/write operations plus code to request access to internal bus, output addresses, release internal bus, etc.

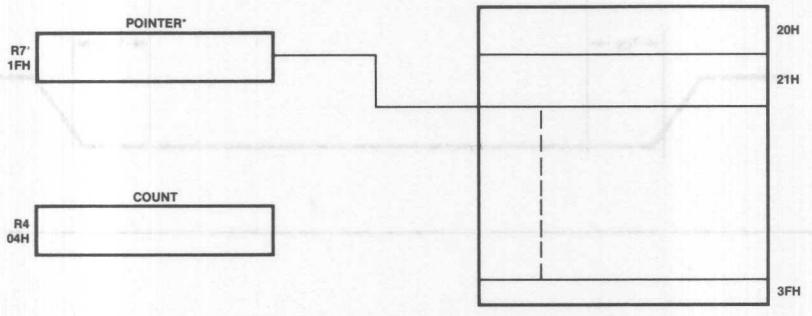
Figure 1 shows how these modules interconnect to implement the 8298 intelligent E^2PROM Controller. Note that the command driver may call the data receiver to get more data. This occurs in the series write and multiple write.

DATA STRUCTURES

Ram Buffer

Used to save buffered data for series write and multiple write commands.

32 bytes dedicated to this function.
— can save 32 data bytes for series write.
— saves 10 address (2 bytes)/ data (1 byte) combinations for multiple write.
— pointer register used to point to next free buffer location.
— count register used to keep track of how many entries are in buffer.



*POINTER IS ACTUALLY DATA DESTINATION REGISTER.

AFN-02027A

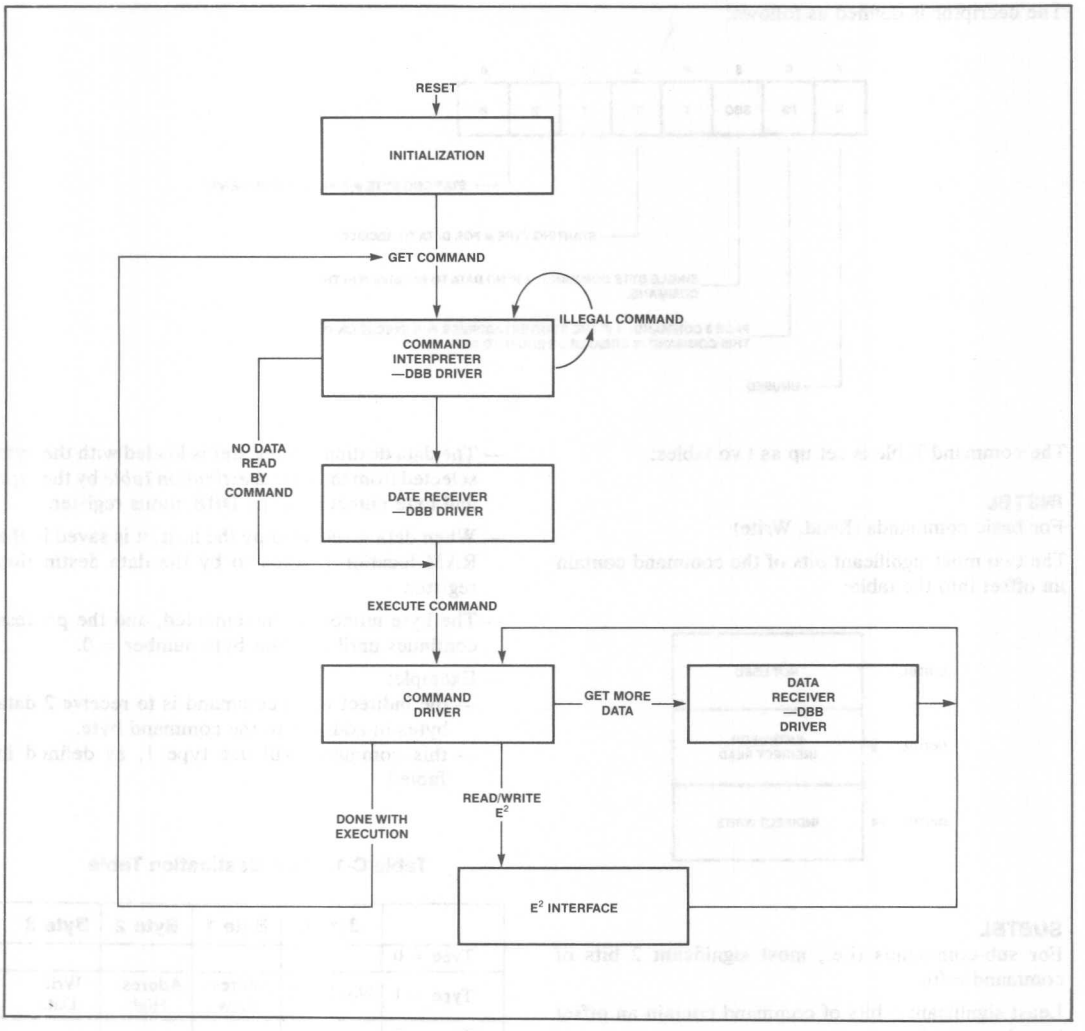
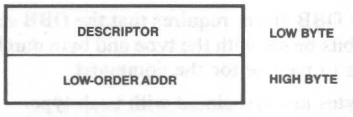


Figure C-1. 8298 Software Execution

Command Table

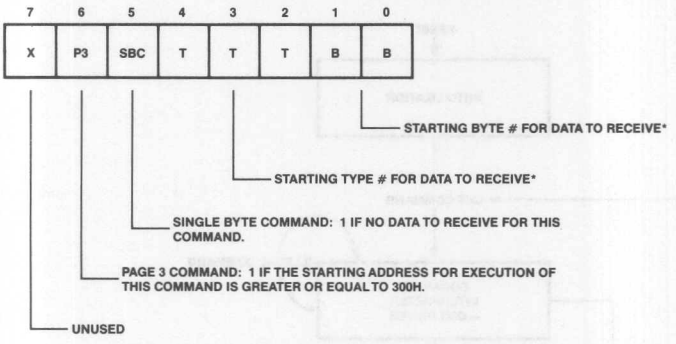
Contains entries for each command. Command table is located in ROM.

Each entry in command table consists of 2 bytes:



where the low order address is the low byte of the starting address for execution of the specified command.

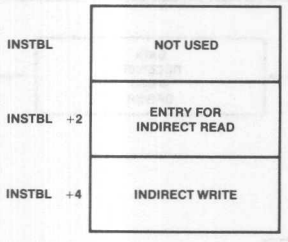
The decriptor is defined as follows:



The command Table is set up as two tables:

INSTBL
For basic commands (Read, Write).

The two most significant bits of the command contain an offset into the table:



SUBTBL
For sub-commands (i.e., most significant 2 bits of command = 0).

Least significant 6 bits of command contain an offset into the sub-table.

Data Destination Table

— Contains pointers to various RAM locations to be loaded with data received for command.

— The DBB driver requires that the DBB status register bits be set with the type and byte numbers of the data to receive for the command.

— 4 bytes are associated with each type.

— 7 bytes are available.

— A routine initializes the DBB status register with the starting byte number and the type of information to get.

— The data destination register is loaded with the byte selected from the *Data Destination Table* by the type and byte numbers in the DBB status register.

— When data is received by the host, it is saved in the RAM location pointed to by the data destination register.

— The byte number is incremented, and the process continues until the 2-bit byte number = 0.

— Example:
  — the indirect write command is to receive 2 data bytes in addition to the command byte.
  — this command will use type 1, as defined in Table 1.

Table C-1. Data Destination Table

	Byte 0	Byte 1	Byte 2	Byte 3
Type = 0				
Type = 1	Not Used	Address Low	Address High	Write Data
Type = 2				

— The descriptor for this command sets Type = 1, Byte = 2.

— The DBB driver will then load RAM with the data bytes received as follows:

1st byte	High address register
2nd byte	Data to Write

— In a similar vein, a routine may initialize the DBB status register with a type and byte number and call the GETDAT subroutine to get up to 4 bytes from the host and save in appropriate RAM locations.

Note: Type 7 is a special type which has the following attributes:

1. The data destination register is *not* loaded from the data destination table. The calling program initializes the data destination register
2. The data destination register is *incremented after* data is received.
3. 1–4 bytes are still received in this manner as selected by the starting byte number.

Figure 2 contains a graphical description of how registers are loaded.

BUS DESCRIPTION OF SOFTWARE ROUTINES

RESET

Entered upon external RESET signal.
a) zeroes: — DBB status register
— E^2 status register
b) initializes default write cycle time.
c) sets 'waiting for command' status.
d) continues w/GETCMD routine.

GETCMD

Called to get a command from user.
a) sets waiting for command bit in DBB status register.
b) optionally sets direct write possible bit
c) exits to GETDBB

CMDCPL

Command Complete—called at completion of command.
a) stops timer/counter in case running.
b) calls RELEAS to release internal bus.
c) enables OBF/IBF external host interrupts.
d) exits to GETCMD

ILLCMD

Illegal Command
a) set 'illegal command' status.
b) exits to GETCMD

DBBI

DBB interrupt service—actually, DBB interrupts are not used, but DBBI is called whenever a command or data is received from the host.
a) disables timer interrupt
b) sets STS = 0

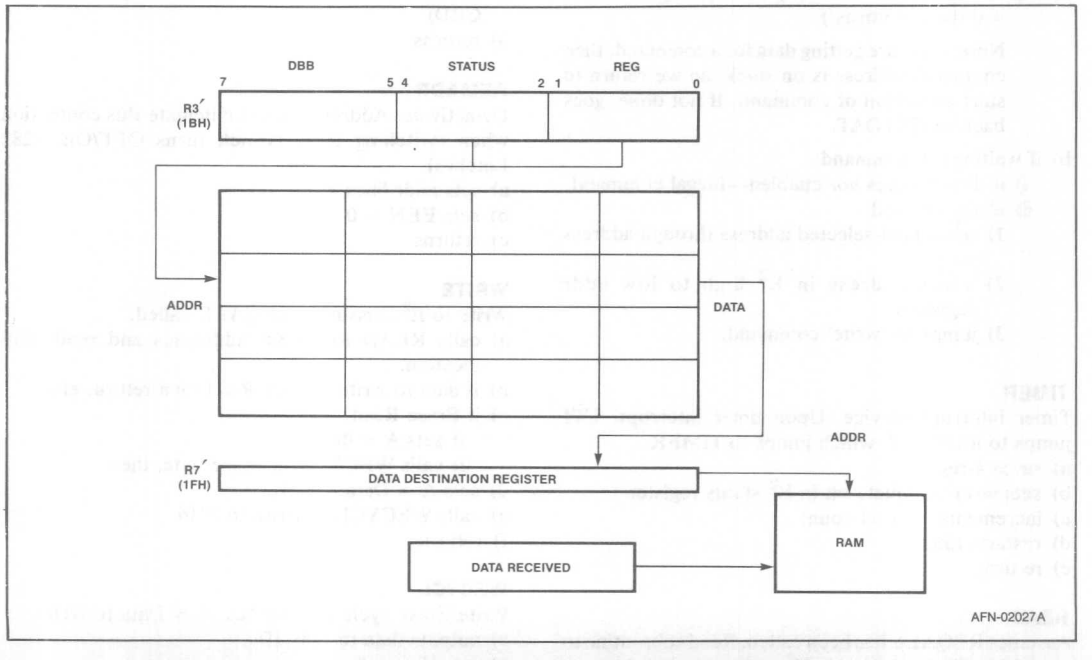


Figure C-2. Set-Up of Data Destination Information

c) determines if command or data received. If *command:*
   i) looks up command in command table, saves descriptor in DBB status register.
   ii) saves command byte in high address register if command is Read or Write.
   iii) saves address of command execution on stack.
   iv) jumps to command execution (via subroutine return) if one-byte command.
   v) continues w/GETDAT routine if *not* one-byte command.
d) if data received, goes to DATRCL

GETDAT
Get Data
a) Looks data destination from R3' (DBB status register) and data destination table. Places address in data destination register (R7).
   — exception: type = 111 —— no look up.
b) Enables external host interrupt.
c) Waits until data received and jumps to DBBI.

DATRCV
If data received from host:
a) if not waiting for command:
   i) saves data in location determined by data destination register (R7).
   ii) increments DBB status, if done, (Byte number = 0 then 'Returns')

   Note: If we are getting data for a command, then command address is on stack, so we return to start execution of command. If not done, goes back to GETDAT.

b) if waiting for command:
   i) if direct writes *not* enabled—illegal command.
   ii) if d/w enabled:
      1) reads host-selected address through address ports.
      2) saves address in E^2 high to low addr registers.
      3) jumps to 'write' command.

TIMER
Timer interrupt service. Upon timer interrupt, UPI jumps to location 7, which jumps to TIMER.
a) saves 4-reg.
b) sets write complete bit in E^2 status register.
c) increments internal count
d) restarts timer
e) returns

READ
Assumes REQALL has been called. Read subroutine to read from 2816. Address in E^2 addr registers. Returns data read in A-Reg.
a) outputs Address (Call OUTADR)
b) set UPI-RD I/O line = 0
c) read data through P1
d) sets UPI-RD I/O line = 1
e) returns

OUTADR
Output addresses from E^2 registers to I/O lines. (1A 0–1A 14)
a) gets high order address.
b) i) shifts MSBs left by 2 so they are in Bits 7, 6
   ii) outputs to Port 2, so IA14 = 27, IA13 = P26.
c) outputs low order address bits
d) outputs bits IA8–IA11.
e) leaves EEN = 1.

REQALL
Request access to local bus and waits until access acknowledged.
a) if access received (T0 = 1) then returns, else:
b) sets EEREQ = 1, EEREQ = 0
c) waits until T0 = 1, (EEACK = 1) and returns,

RELEAS
Release control of local bus
a) sets all Address lines = 1
b) sets all data lines = 1
c) de-activates all control lines (EEREQ, EEN, VPP, CRD)
d) returns

REMADR
De-activates Address lines to eliminate Bus contention when switching EEN (which turns OFF/On, 8282 Latches)
a) sets addr lines = 1
b) sets EEN = 0
c) returns

WRITE
Write to E^2—assumes REQALL called.
a) calls READ to output addresses and read 2816 location.
b) if data to write = Data Read then return, else
c) if Erase Read, then:
   i) sets A = 0FFH
   ii) calls WECYCL to Erase byte, then
d) sets A = Data to Write
e) calls WECYCL to write to 2816
f) returns

WECYCL
Write/Erase cycle subroutines. A = Data to Write.
a) outputs data to ID0–ID7.
b) sets V_{PP} = 0
c) resets write complete bit of E^2 status register.

APPENDIX C: 8298 FIRMWARE DESCRIPTION

d) starts timer
e) waits until write complete bit set by timer interrupt service routine. Calls CKDBB to check for abort command while waiting.
f) at end of write cycle, calls SHUT to shut down V_{PP} switch
g) returns

SHUT
Turns off V_{PP} switch
a) set V_{PP} = 0, UPP = 1
b) waits a time of 100 μs for V_{PP} signal to fall to 5V
c) returns

CKDBB
Check data Bus Buffer for abort command.
a) If buffer not full THEN returns ELSE
If not command in buffer (F1 = 0)
THEN returns
ELSE If abort command THEN GOTO abort routine
ELSE — set illegal command status
— return

INCADR
Increments E^2 high, low address register pair.

DECCNT
Decrements count register pair. If zero, returns with REG = 0

Command Driver

The following are descriptions of routines which execute commands. When these routines are called, data for the selected command has already been received.

READC
Read Command
a) Requests access to local bus (REQACC)
b) Calls read Subroutine
c) Calls OUTPUT to output result to host

OUTPUT
Called to place data in A-Reg into output buffer. Waits until host has read output buffer before exiting.

WRITEC
Write Command
a) calls REQACC to request access to local bus.
b) calls write subroutine

CEREASE
Chip Erase Command
a) requests access to bus
b) moves high order address bits, IAB, IA14 to P26, P27.
c) outputs high order address to P1 with bit 7 = 0 (CERASE) = 0
d) pulses EEN to V_{OH} to latch high address and chip erase signal.
e) calls WECYCL with A-REG = 0FFH to erase a chip
f) calls REMCE to remove chip erase signal

REMCE
Remove chip erase
a) sets P1 = 0FFH
b) sets EEN = 0
c) sets EEN = 1 to latch. No chip erase
d) exits

BLOCKE
Block erase command
a) requests access
b) sets data to write = 0FFH (erase)
c) writes to E^2
d) remove address to prevent bus contention
e) increment address
f) decrements count, if not done go to step b)

MULTWR
Multiple write command
a) zeroes buffer count and initializes buffer pointer
b) calls GETDAT to get 10 address/data combinations or # combinations left to get, whichever is lower.
— Data is loaded into RAM buffer
c) dumps buffer to E^2.
Note: Data is saved in RAM as follows:
High Addr
Low Addr
Data to Write
d) continues to step b) if not done

SERWRT
Series write command
a) zeroes buffer counter and initializes buffer pointer.
b) gets up to 32 bytes of data or # bytes left to get, whichever is less.
c) requests access, dumps data in buffer to E^2 memory. Addresses are incremented after each byte is written.
d) continues to step b) if not done

SERRD
Series read command
a) requests access to bus
b) reads a byte
c) outputs it to host
d) increments address
e) decrements count
f) continues to step b) if not done

APPENDIX C: 8298 FIRMWARE DESCRIPTION

ABORT

a) If V_{PP} is on (*UPP* = 0)
   i) calls SHUT to turn off V_{PP}.
b) calls REMCE to remove chip erase in case activated.

READAL

Read last low address.

READAH

Read last high address.

RADWR

Read last data to write to E^2.

APPENDIX D: 8298 E^2PROM CONTROLLER FIRMWARE LISTING

```
LOC  OBJ        LINE      SOURCE STATEMENT
                   1 $MOD41A MACROFILE DEBUG
                   2
                   3
                   4
                   5         ;
                   6         ;
                   7         ;
                   8         ;
                   9         ;
                  10         ;
                  11         ;
                  12         ;
                  13         ;
                  14         ;
                  15         ;
                  16
                  17
                  18
                  19
                  20         ;       UPI REGISTER DEFINITIONS
                  21
                  22         ;       RB0:
                  23
0000              24 SCR0     EQU     0       ; R0 = SCRATCH
0001              25 SCR1     EQU     1       ; R1 = SCRATCH
0002              26 SCR2     EQU     2       ; R2 = SCRATCH
0003              27 INTCNT   EQU     3       ; R3 = INTERVAL COUNT
0004              28 BUFCNT   EQU     4       ; R4 = BUFFER COUNTER
0005              29 WRTDAT   EQU     5       ; R5 = DATA TO WRITE
0006              30 CNTLO    EQU     6       ; R6 = LOW ORDER COUNT
0007              31 CNTHI    EQU     7       ; R7 = HIGH ORDER COUNT
                  32
                  33
                  34         ;       RB1:
                  35
0018              36 SCR0P    EQU     18H     ; R0' = SCRATCH
0019              37 SCR1P    EQU     19H     ; R1' = SCRATCH
001A              38 COMFB    EQU     1AH     ; R2' = COMMAND FIRST BYTE
001B              39 DBSTAT   EQU     1BH     ; R3' = DBB STATUS REGISTER
001C              40 ASAVE    EQU     1CH     ; R4' = A-REGISTER SAVE
001D              41 EESTAT   EQU     1DH     ; R5' = EE STATUS REGISTER
001E              42 INTIME   EQU     1EH     ; R6' = INITIAL WIRTE TIMER COUNT
001F              43 DATDES   EQU     1FH     ; R7' = DATA DESTINATION REGISTER
                  44
                  45 $EJECT
```

APPENDIX D: 8298 E²PROM CONTROLLER FIRMWARE LISTING

```
LOC  OBJ        LINE      SOURCE STATEMENT
                  46
                  47        ;       NON-REGISTER MEMORY DEFINITIONS
                  48
0014              49 ADRLO  EQU     14H     ; LOW ORDER EE ADDRESS
0015              50 ADRHI  EQU     15H     ; HI ORDER EE ADDRESS
0020              51 BUFST  EQU     20H     ; BUFFER START ADDRESS
0016              52 ININT  EQU     16H     ; INITIAL INTERVAL COUNT
003E              53 MWBEND EQU     20H+30D ; MULTIPLE WRITE BUFFER END ADDRESS
0040              54 SWBEND EQU     20H+32D ; SERIAL WRITE BUFFER END ADDRESS
                  55
                  56
                  57        ;       I/O DEFINITIONS
                  58
                  59        ;       HOST INTERRUPTS
                  60
00DF              61 IBFINA EQU     11011111B       ; INACTIVE IBF' HOST INTERRUPT
FF20              62 IBFACT EQU     NOT IBFINA      ; ACTIVE IBF' HOST INTERRUPT
0010              63 OBFACT EQU     00010000B       ; ACTIVE OBF HOST INTERRUPT
FFEF              64 OBFINA EQU     NOT OBFACT      ; INACTIVE OBF HOST INTERRUPT
                  65
                  66        ;       CONTROL LINES
                  67
000E              68 VPPACT EQU     1110B           ; ACTIVE VPP SWITCH
FFF1              69 VPPINA EQU     NOT VPPACT      ; INACTIVE VPP SWITCH
0007              70 REQACT EQU     0111B           ; EEREQ ACTIVE SIGNAL
0009              71 RDACT  EQU     1001B           ; ACTIVATE UPIRD' AND EEN'
0002              72 RDINA  EQU     0010B           ; DE-ACTIVATE UPIRD'
000B              73 EENACT EQU     1011B           ; ACTIVATE EEN'
0004              74 EENINA EQU     0100B           ; DEACTIVATE EEN'
                  75
                  76        ;       STATUS DEFINITIONS
                  77
0007              78 WIPSTS EQU     0111B           ; WRITE-IN-PROGRESS STATUS
                  79
                  80
                  81        ;       TIME VALUES
                  82
FF83              83 TIME   EQU     -125D           ; TIMER COUNT VALUE FOR 3.333 MSEC TIME COUNT
007F              84 INITWR EQU     7FH             ; INIT EESTAT WRT VALUE FOR WRT COMPL BIT =0
000E              85 VPPFALL EQU     14D            ; VPP FALL TIME.  SET FOR > 100 USEC
                  86                                ; FOR WRITE CYCLE TIMER COUNT
                  87
                  88
                  89        ;       MISC
                  90
000A              91 ABORTC EQU     0AH             ; ABORT COMMAND CODE
                  92
                  93 $EJECT
```

APPENDIX D: 8298 E^2PROM CONTROLLER FIRMWARE LISTING

```
LOC  OBJ        LINE      SOURCE STATEMENT
                   94
                   95 ;              REGISTER FORMATS:
                   96
                   97
                   98 ;        1) DBB STATUS REGISTER (R3' = 1BH)
                   99 ;
                  100 ;                 BIT(S)          DESCRIPTION
                  101 ;                 ------          -----------
                  102 ;                 0-1             NEXT BYTE # TO GET (0-3)
                  103 ;                 2-4             CURRENT TYPE OF BYTE (FROM DATA DEST TABLE)
                  104 ;                  5              WAITING FOR COMMAND (=1)
                  105 ;                  6              DIRECT WRITE ENABLED (=1)
                  106 ;
                  107 ;
                  108 ;        2) EE STATUS REGISTER (R5' = 1DH)
                  109 ;
                  110 ;                 BIT(S)          DESCRIPTION
                  111 ;                 ------          -----------
                  112 ;                 0-3             INTERVAL LOW ORDER COUNT (0-3)
                  113 ;                  7              WRITE COMPLETE INTERRUPT RECEIVED
                  114 ;
                  115 ;
                  116 ;        3) F0 = DIRECT WRITE POSSIBLE
                  117 ;
                  118 ;        4) FLAG F1 = COMMAND RECEIVED (=1)
                  119 ;
                  120
                  121 $EJECT
```

APPENDIX D: 8298 E^{2}PROM CONTROLLER FIRMWARE LISTING

```
LOC  OBJ        LINE        SOURCE STATEMENT
                  122
                  123         ;       INITIALIZATION
                  124
0000              125         ORG     0
                  126
0000 4609         127 RESET:  JNT1    INIT
                  128
0007              129         ORG     7H
0007 04FC         130         JMP     TIMER           ; TIMER INTERRUPT VECTOR
                  131
                  132 INIT:
0009 F5           133         EN      FLAGS           ; ENABLE HOST INTERRUPT FLAGS
000A 85           134         CLR     F0              ; SET F0 = 0 TO INDICATE DIRECT WRITE NOT POSS
000B D5           135         SEL     RB1             ; SELECT ALTERNATE REG SET TEMPORARILY
000C 27           136         CLR     A               ; SET A = 0
000D AB           137         MOV     R3,A            ; ZERO DBB STATUS REGISTER
000E AD           138         MOV     R5,A            ; ZERO EE STATUS REGISTER
000F BE83         139         MOV     R6,#TIME        ; INITIAL WRITE TIMER VALUE
0011 C5           140         SEL     RB0             ; SELECT NORMAL REGISTER SET
0012 23F0         141         MOV     A,#0F0H         ; OUTPUT WAITING FOR COMMAND STATUS
0014 90           142         MOV     STS,A
                  143
                  144 GETCMD:
0015 A5           145         CLR     F1              ; RESET LAST STATE OF COMMAND/DATA REGISTER
0016 D5           146         SEL     RB1             ; SELECT ALTERNATE REG BANK
0017 FB           147         MOV     A,R3            ; GET DBB STATUS
0018 4320         148         ORL     A,#20H          ; OR IN WAITING FOR COMMAND BIT
001A 85           149         CLR     F0              ; CLEAR DIRECT WRITE POSSIBLE BIT
001B D21F         150         JB6     DWE             ; ONLY SET D/W POSSIBLE IF D/W ENABLED
001D 0420         151         JMP     DWNE            ; D/W NOT ENABLED -DON'T SET BIT
                  152 DWE:
001F 95           153         CPL     F0              ; SET D/W POSSIBLE BIT
                  154 DWNE:
0020 AB           155         MOV     R3,A            ; RE-SAVE DBB STATUS REG
0021 04B3         156         JMP     GETDBB          ; GO GET SOME MORE DATA
                  157
                  158
                  159         ;       COMMAND COMPLETE
                  160
                  161 CMDCPL:
0023 C5           162         SEL     RB0             ; SELECT NORMAL REG SET
0024 65           163         STOP    TCNT            ; STOP ANY TIMER ACTIVITY
0025 3430         164         CALL    RELEAS          ; RELEASE CONTROL OF BUS
0027 23F0         165         MOV     A,#0F0H         ; SET COMMAND COMPLETE STATUS
0029 90           166         MOV     STS,A
002A 8A30         167         ORL     P2,#00110000B   ; ALLOW OBF,IBF HOST INTERRUPTS
002C 0415         168         JMP     GETCMD          ; GO GET ANOTHER COMMAND
                  169
                  170         ;       ILLEGAL COMMAND:
                  171
                  172 ILLCMD:
002E 23E0         173         MOV     A,#11100000B    ; SET ILLEGAL COMMAND STATUS
0030 90           174         MOV     STS,A
0031 0415         175         JMP     GETCMD
                  176
```

APPENDIX D: 8298 E²PROM CONTROLLER FIRMWARE LISTING

```
LOC  OBJ        LINE       SOURCE STATEMENT
                 178
                 179        ;       DBB INTERRUPT SERVICE ROUTINE AND DATA RETRIEVER
                 180
                 181 DBBI:
0033 35          182        DIS     TCNTI           ; DISABLE INTERRUPTS
0034 27          183        CLR     A               ; ZERO STS REG BECAUSE WE'RE NO LONGER LOOKING
                 184                                ; FOR COMMAND/DATA
0035 90          185        MOV     STS,A
0036 D5          186        SEL     RB1             ; SELECT THIS REGISTER SET
0037 AC          187        MOV     R4,A            ; SAVE A-REGISTER
0038 9ADF        188        ANL     P2,#IBFINA      ; DE-ACTIVATE IBF INTERRUPTS TO HOST
003A 85          189        CLR     F0              ; CLEAR DWP BIT
003B 763F        190        JF1     DBBI07          ; IF COMMAND RECEIVED, CONTINUE, ELSE
003D 04C7        191        JMP     DATRCV          ; COMMAND - GO PROCESS THE COMMAND
                 192 DBBI07:
003F 22          193        IN      A,DBB           ; GET COMMAND
0040 AA          194        MOV     R2,A            ; SAVE IT
0041 53F0        195        ANL     A,#0F0H         ; CHECK IF ILLEGAL COMMAND (30H - 3FH)
0043 D24D        196        JB6     NOTICD          ; TO SAVE SPACE, PART OF TABLE ELIMINATED
0045 F24D        197        JB7     NOTICD          ; MUST HAVE B5 AND B4 = 1 WITH B6 AND B7
0047 924B        198        JB4     MAYBIC          ; = 0 TO HAVE AN ILLEGAL COMMAND.
0049 044D        199        JMP     NOTICD
                 200 MAYBIC:
004B B22E        201        JB5     ILLCMD          ; YES, IT IS ILLEGAL. GO DEAL WITH IT
                 202 NOTICD:
004D 53C0        203        ANL     A,#0C0H         ; MASK ALL BUT HI ORDER BITS
004F C675        204        JZ      SUBCMD          ; IF HI ORDER BITS ZERO THEN SUB COMMAND
0051 E7          205        RL      A               ; MOVE HI BITS TO LOW BITS
0052 E7          206        RL      A               ; AND MULTIPLY BY 2 IN THE PROCESS
0053 E7          207        RL      A
0054 0365        208        ADD     A,#LOW INSTBL   ; LET A POINT TO ENTRY IN INSTRUCTION TABLE
                 209 LOOKUP:
0056 A8          210        MOV     R0,A            ; SAVE POINTER TEMPORARILY
0057 E3          211        MOVP3   A,@A            ; GET COMMAND DESCRIPTOR BYTE FROM TABLE
0058 28          212        XCH     A,R0            ; SWAP DESCRIPTOR FOR TABLE ADDRESS
0059 17          213        INC     A               ; POINT TO COMMAND START ADDRESS IN TABLE
005A E3          214        MOVP3   A,@A            ; GET COMMAND START ADDRESS
005B 28          215        XCH     A,R0            ; SWAP DEST ADDR FOR COMMAND DESCRIPTOR
005C A9          216        MOV     R1,A            ; SAVE DESCRIPTOR IN R1
005D 531F        217        ANL     A,#1FH          ; MASK TO KEEP DATA TO RECEIVE
005F 2B          218        XCH     A,R3            ; SWAP A & DBB STATUS
0060 53E0        219        ANL     A,#0E0H         ; REMOVE LOW ORDER BITS
0062 4B          220        ORL     A,R3            ; ADD IN DATA DESCRIPTOR
0063 AB          221        MOV     R3,A            ; SAVE UPDATED DBB REG
0064 B27D        222        JB5     PCMND           ; IF WAITING FOR COMMAND BIT SET GO PROCESS IT
0066 23F0        223        MOV     A,#-1*(LOW ABORT); WE'RE NOT WAITING FOR COMMAND, SO THE ONLY
                 224                                ; COMMAND WE'LL ACCEPT IS ABORT - CHECK FOR
                 225                                ; ABORT COMMAND
0068 68          226        ADD     A,R0
0069 966D        227        JNZ     ILLCDO          ; NOT ABORT - GO INDICATE ILLEGAL COMMAND
006B 6410        228        JMP     ABORT           ; YES ABORT - GO PROCESS IT
                 229
                 230        ;       ILLEGAL COMMAND DURING WRITE CYCLE
                 231
                 232 ILLCDO:
```

APPENDIX D: 8298 E²PROM CONTROLLER FIRMWARE LISTING

```
LOC  OBJ       LINE      SOURCE STATEMENT
006D 23AA       233       MOV     A,#0AAH         ; OUTPUT ILLEGAL COMMAND MESSAGE
006F 02         234       OUT     DBB,A
0070 8910       235       ORL     P1,#0BFACT      ; ALLOW OBF EXTERNAL INTERUPTS
                236 EXIT:
0072 FC         237       MOV     A,R4            ; RESTORE A-REGISTER
0073 25         238       EN      TCNTI           ; RE-ENABLE TIMER INTERRUPT
0074 93         239       RETR                    ; RETURN TO USER PROGRAM
                240
                241       ;       LOOK UP SUB COMMAND AND POINT TO ENTRY IN TABLE
                242
                243 SUBCMD:
0075 FA         244       MOV     A,R2            ; GET COMMAND
0076 533F       245       ANL     A,#3FH          ; KEEP LOWER 6 BITS
0078 E7         246       RL      A               ; MULTIPLY BY 2 BECAUSE 2 BYTES PER INSTR
0079 036D       247       ADD     A,#LOW SUBTBL   ; ADD START ADDR OF SUB COMMAND TABLE
007B 0456       248       JMP     LOOKUP          ; GO LOOKUP SUB COMMAND
                249
                250
                251       ;       PROCESS THE COMMAND
                252
                253 PCMND:
007D A5         254       CLR     F1              ; CLEAR COMMAND FLAG
007E 27         255       CLR     A               ; ZERO STS BITS
007F 90         256       MOV     STS,A
                257 PC03:
0080 FB         258       MOV     A,R3            ; GET DBB STATUS
0081 53DF       259       ANL     A,#11011111B    ; CLEAR WAITING FOR COMMAND BIT
0083 AB         260       MOV     R3,A            ; RE-SAVE DBB STATUS
0084 C7         261       MOV     A,PSW           ; GET PSW
0085 2311       262       MOV     A,#00010001B    ; SET STACK POINTER = 1ST LOC
0087 D7         263       MOV     PSW,A           ; AND RESTORE PSW
0088 FA         264       MOV     A,R2            ; GET COMMAND FIRST BYTE IN A
0089 F2B9       265       JB7     SAVEHI          ; IF BIT 7 OR BIT 6 SET SAVE HI ADDRESS
008B D2B9       266       JB6     SAVEHI
                267 PC05:
008D F8         268       MOV     A,R0            ; SET COMMAND DEST ADDRESS IN STACK
008E B808       269       MOV     R0,#8H          ; IN 1ST LOCATION IN STACK
0090 A0         270       MOV     @R0,A
0091 F9         271       MOV     A,R1            ; GET COMMAND DESCRIPTOR
0092 D298       272       JB6     P3INST          ; IF PAGE 3 BIT SET ADD PAGE 3 ADDRESS
0094 2302       273       MOV     A,#HIGH FIRSTIN ; SET HI STACK BYTE FOR RB0 SELECTED
                274                               ; AND HIGH ORDER ADDRESS OF 1ST INSTRUCTION
0096 049A       275       JMP     PC10
                276 P3INST:
0098 2303       277       MOV     A,#HIGH FIRSTP3 ; SET HI STACK BYTE FOR RB0 SELECTED
                278                               ; AND HI ORDER ADDRESS OF 1ST PAGE 3 INSTR
                279 PC10:
009A 18         280       INC     R0              ; POINT TO HIGH ORDER STACK LOCATION
009B A0         281       MOV     @R0,A           ; SAVE ON STACK
009C F9         282       MOV     A,R1            ; GET COMMAND DESCRIPTOR
009D B2C3       283       JB5     EXEC            ; GO EXECUTE COMMAND IF NO DATA TO RECEIVE
                284
                285       ;       GET DATA (USER PROGRAM MAY ENTER HERE)
                286
                287 GETDAT:
```

APPENDIX D: 8298 E^2PROM CONTROLLER FIRMWARE LISTING

```
LOC  OBJ        LINE       SOURCE STATEMENT
009F B900        288        MOV     R1,#0           ; ZERO STS DATA COUNTER
                 289 SETGET:
00A1 F9          290        MOV     A,R1            ; GET STS DATA COUNT
00A2 4308        291        ORL     A,#8H           ; OR-IN WAITING FOR COMMAND/DATA BIT
00A4 47          292        SWAP    A               ; MOVE TO HIGH ORDER BITS
00A5 90          293        MOV     STS,A           ; AND OUTPUT TO STS REGISTER
00A6 FB          294        MOV     A,R3            ; GET DBB STATUS
00A7 43E3        295        ORL     A,#11100011B    ; CHECK FOR TYPE = 111
00A9 37          296        CPL     A               ; IF TYPE = 111 THEN A-REG SHOULD BE 0
00AA C6B3        297        JZ      GETDBB          ; IF SPECIAL TYPE (=111) DON'T LOOK UP DEST
00AC FB          298        MOV     A,R3            ; GET DBB STATUS
00AD 531F        299        ANL     A,#1FH          ; KEEP ONLY TYPE AND BYTE BITS
00AF 03CD        300        ADD     A,#LOW DESTBL   ; ADD START OF DESTINATION TABLE
00B1 E3          301        MOVP3   A,@A            ; GET DESTINATION
00B2 AF          302        MOV     R7,A            ; AND SAVE IN DATA DESTINATION REGISTER
                 303 GETDBB:
00B3 8A20        304        ORL     P2,#IBFACT      ; ENABLE HOST INTERRUPT
00B5 D6B5        305 WAITB:  JNIBF   WAITB           ; WAIT UNTIL DATA RECEIVED
00B7 0433        306        JMP     DBBI            ; GO PROCESS DATA WE JUST GOT
                 307
                 308        ;       SAVE HI-ORDER ADDRESS FOR READ OR WRITE
                 309
                 310 SAVEHI:
00B9 29          311        XCH     A,R1            ; SAVE R1 IN R7
00BA 2F          312        XCH     A,R7
00BB B915        313        MOV     R1,#ADRHI       ; POINT TO HI ORDER EE ADDRESS
00BD FA          314        MOV     A,R2            ; GET COMMAND FIRST BYTE TO PUT IN HI ADDRESS
00BE A1          315        MOV     @R1,A           ; SAVE COMMAND IN HI ORDER ADDRESS REGISTER
00BF 2F          316        XCH     A,R7            ; RESTORE R1
00C0 A9          317        MOV     R1,A
00C1 048D        318        JMP     PC05            ; CONTINUE PROCESSING COMMAND
                 319
                 320        ;       EXECUTE COMMAND
                 321
                 322 EXEC:
00C3 27          323        CLR     A               ; CLEAR STS TO INDICATE PROCESSING
00C4 90          324        MOV     STS,A
00C5 0472        325        JMP     EXIT            ; AND EXIT.
                 326
                 327        ;       DATA RECEIVED
                 328
                 329 DATRCV:
00C7 22          330        IN      A,DBB           ; GET DBB DATA
00C8 A8          331        MOV     R0,A            ; SAVE TEMPORARILY IN R0
00C9 FB          332        MOV     A,R3            ; GET DBB STATUS
00CA B2DD        333        JB5     DWCMD           ; IF WE'RE WAITING FOR COMMAND THEN MAYBE D/W
00CC 2F          334        XCH     A,R7            ; SAVE A IN R7 AND GET DATA DEST PTR
00CD 28          335        XCH     A,R0            ; GET DATA TO SAVE IN A & ADDR IN R0
00CE A0          336        MOV     @R0,A           ; SAVE IN APPROPRIATE LOCATION
00CF 28          337        XCH     A,R0            ; RETRIEVE ADDRESS IN A
00D0 17          338        INC     A               ; INCREMENT DESTINATION ADDRESS IN CASE OF SPEC
                 339                                ; TYPE
00D1 2F          340        XCH     A,R7            ; SAVE IN R7 & RETRIEVE DBB IN A
00D2 17          341        INC     A               ; INCREMENT BYTE # TO GET
00D3 53DF        342        ANL     A,#11011111B    ; MAKE SURE WAITING FOR COMMAND BIT = 0
```

APPENDIX D: 8298 E²PROM CONTROLLER FIRMWARE LISTING

```
LOC  OBJ        LINE        SOURCE STATEMENT
00D5 AB          343          MOV     R3,A            ; RE-SAVE DBB STATUS
00D6 5303        344          ANL     A,#3H           ; MASK OUT ALL BUT BYTE COUNT
00D8 C6C3        345          JZ      EXEC            ; IF ZERO THE WE'RE DONE - CONTINUE CMMND EXEC
00DA 19          346          INC     R1              ; INCREMENT DATA BYTE COUNTER
00DB 04A1        347          JMP     SETGET          ; NO - GO GET MORE DATA
                 348
                 349          ;       DIRECT WRITE COMMAND
                 350
                 351 DWCMD:
00DD D2E1        352          JB6     DW05            ; IF DIRECT WRITES ENABLED THEN GO PROCESS
00DF 042E        353          JMP     ILLCMD          ; ELSE ILLEGAL COMMAND RECEIVED
                 354
                 355 DW05:
00E1 2305        356          MOV     A,#WRTDAT       ; POINT TO WRITE DATA REGISTER
00E3 28          357          XCH     A,R0            ; SWAP
00E4 A0          358          MOV     @R0,A           ; SAVE DATA TO WRITE IN REGISTER
00E5 B814        359          MOV     R0,#ADRLO       ; POINT TO LOW ORDER ADDRESS REGISTER
00E7 8D          360          MOVD    A,P5            ; GET LOW ORDER ADDRESS FROM ADDRESS PORTS
00E8 47          361          SWAP    A
00E9 A9          362          MOV     R1,A            ; SAVE TEMPORARILY IN R1
00EA 8C          363          MOVD    A,P4
00EB 49          364          ORL     A,R1            ; BRING IN HI ORDER NIBBLE
00EC A0          365          MOV     @R0,A           ; SAVE IN LOW ORDER ADDRESS REGISTER
00ED 8E          366          MOVD    A,P6            ; GET HIGH ORDER ADDRESS
00EE 18          367          INC     R0              ; POINT TO HI ORDER ADDRESS
00EF A0          368          MOV     @R0,A           ; SAVE HI ORDER ADDRESS
00F0 8A          369          IN      A,P2            ; GET HI ORDER ADDRESS BITS
00F1 77          370          RR      A               ; MOVE TWO BITS TO THE RIGHT
00F2 77          371          RR      A
00F3 5330        372          ANL     A,#30H          ; MAKE SURE THAT'S ALL THE DATA WE HAVE
00F5 40          373          ORL     A,@R0           ; READ IN REST OF HI ORDER ADDRESS BYTE
00F6 A0          374          MOV     @R0,A           ; AND SAVE IN HI ORDER ADDRESS REGISTER
                 375 DW10:
00F7 2300        376          MOV     A,#00000000B    ; SET STACK PROINTER TO POINT TO
                 377                                  ; 2ND LEVEL STACK AND RB0 SELECTED
00F9 D7          378          MOV     PSW,A
00FA 4410        379          JMP     WRITEC          ; GO EXECUTE WRITE COMMAND
                 380
                 381 $EJECT
```

APPENDIX D: 8298 E^{2}PROM CONTROLLER FIRMWARE LISTING

```
LOC  OBJ        LINE       SOURCE STATEMENT
                 382
                 383        ;       TIMER INTERRUPT
                 384
                 385 TIMER:
00FC D5          386        SEL     RB1             ; SELECT ALTERNATE REGISTER BANK
00FD AC          387        MOV     R4,A            ; SAVE A-REG
00FE 1D          388        INC     R5              ; INCREMENT INTERVAL COUNT
00FF FD          389        MOV     A,R5            ; GET EE STATUS
0100 4380        390        ORL     A,#80H          ; SET WRITE COMPLETE INTERRUPT BIT
0102 AD          391        MOV     R5,A            ; RESTORE EE STATUS
0103 FE          392        MOV     A,R6            ; RESET TIMER COUNT
0104 62          393        MOV     T,A
0105 FC          394        MOV     A,R4            ; RESTORE A-REG
0106 93          395        RETR                    ; BACK TO INTERUPTED PROGRAM
                 396
                 397
                 398
                 399        ;       READ / WRITE SUBROUTINES
                 400
                 401
                 402
                 403        ;       ************************************************
                 404
                 405        ;               EE READ - CALLED W/ ADDRESS IN ADDRESS REG
                 406
                 407        ;               DATA RETURNED IN A-REG
                 408        ;               USES:   R0
                 409        ;               ENABLES LOCAL BUS
                 410
                 411        ;               ASSUMES ACCESS TO LOCAL BUS
                 412
                 413        ;       ************************************************
                 414
                 415 READ:
0107 3413        416        CALL    OUTADR          ; OUTPUT EE ADDRESS
0109 2309        417        MOV     A,#RDACT        ; ACTIVATE READ LINE
010B 9F          418        ANLD    P7,A
010C 09          419        IN      A,P1            ; READ THE DATA
010D A8          420        MOV     R0,A            ; SAVE TEMPORARILY
010E 2302        421        MOV     A,#RDINA        ; DE-ACTIVATE READ LINE
0110 8F          422        ORLD    P7,A            ; BUT LEAVE EEN ACTIVE
0111 F8          423        MOV     A,R0            ; RESTORE A
0112 83          424        RET
                 425
                 426
                 427        ;       ************************************************
                 428
                 429        ;               OUTADR - OUTPUT ADDRESSES TO ADDRESS LINES
                 430
                 431        ;               USES A-REG, R0
                 432
                 433        ;       ************************************************
                 434
                 435 OUTADR:
0113 B815        436        MOV     R0,#ADRHI       ; POINT TO HI ORDER ADDRESS REGISTER
```

APPENDIX D: 8298 E^{2}PROM CONTROLLER FIRMWARE LISTING

```
LOC  OBJ        LINE        SOURCE STATEMENT
0115 F0          437         MOV     A,@R0           ; GET HI ORDER ADDRESS
                 438
                 439         ;       HI ORDER ADDRESS OUTPUT (OUTPUT A12,A13 TO P26,P27)
                 440
0116 E7          441         RL      A               ; MOVE A12,A13 TO BITS 6,7
0117 E7          442         RL      A
0118 53C0        443         ANL     A,#0C0H         ; MAKE SURE ONLY BITS 6,7 SET
011A 3A          444         OUTL    P2,A            ; AND OUTPUT TO PORT 2
011B 230B        445         MOV     A,#EENACT       ; ACTIVATE EEN'
011D 9F          446         ANLD    P7,A
                 447
                 448 OUTA20:
011E B814        449         MOV     R0,#ADRLO       ; POINT TO LOW ORDER ADDRESS REGISTER
0120 F0          450         MOV     A,@R0           ; GET LOW ORDER ADDRESS
0121 3C          451         MOVD    P4,A            ; OUTPUT LOW ORDER ADDRESS
0122 47          452         SWAP    A
0123 3D          453         MOVD    P5,A
0124 18          454         INC     R0              ; POINT TO HI ORDER ADDRESS
0125 F0          455         MOV     A,@R0           ; GET HI ORDER ADDRESS
0126 3E          456         MOVD    P6,A            ; AND OUTPUT IT
0127 83          457         RET                     ; BACK TO CALLING PROGRAM
                 458
                 459
                 460
                 461         ;       ************************************************
                 462
                 463         ;               REQACC - REQUEST ACCESS TO BUS AND
                 464         ;                        SIEZE BUS
                 465
                 466         ;               USES:   A-REG
                 467
                 468         ;       ************************************************
                 469
                 470 REQACC:
0128 362F        471         JT0     GOTACC          ; IF ALREADY GOT ACCESS SKIP REQUEST
012A 2307        472         MOV     A,#REQACT       ; REQUEST ACCESS
012C 9F          473         ANLD    P7,A
012D 262D        474 WAITAC: JNT0    WAITAC          ; WAIT FOR ACCESS
                 475 GOTACC:
012F 83          476         RET                     ; BACK TO CALLER
                 477
                 478
                 479         ;       ************************************************
                 480
                 481         ;               RELEASE - RELEASE BUS
                 482
                 483         ;               USES:   A-REG
                 484
                 485         ;       ************************************************
                 486
                 487 RELEAS:
0130 23FF        488         MOV     A,#0FFH         ; WRITE 1'S TO ALL ADDR AND DATA LINES
0132 39          489         OUTL    P1,A
0133 3C          490         MOVD    P4,A
0134 3D          491         MOVD    P5,A
```

APPENDIX D: 8298 E^{2}PROM CONTROLLER FIRMWARE LISTING

```
LOC  OBJ        LINE        SOURCE STATEMENT
0135 3E          492         MOVD    P6,A
0136 8AC0        493         ORL     P2,#0C0H        ; SET P26,P27 = 1
                 494
0138 3F          495 RELRET: MOVD    P7,A
0139 83          496         RET                     ; RETURN TO CALLER
                 497
                 498
                 499         ;       ***********************************************
                 500
                 501         ;               REMADR - REMOVE ADDRESSES
                 502         ;                        (SET ADDRESS LINES = 1)
                 503         ;                        ALSO DEACTIVATES EEN' (SETS TO 1)
                 504
                 505         ;               USES:   A-REG
                 506
                 507         ;       ***********************************************
                 508
                 509 REMADR:
013A 23FF        510         MOV     A,#0FFH                 ; SET A = ALL 1'S
013C 8C          511         ORLD    P4,A                    ; OUTPUT TO A8-A11
013D 8D          512         ORLD    P5,A
013E 8E          513         ORLD    P6,A
013F 8AC0        514         ORL     P2,#0C0H
                 515
0141 2304        516 REMRET: MOV     A,#EENINA               ; DE-ACTIVATE 2816'S
0143 8F          517         ORLD    P7,A
0144 83          518         RET
                 519
                 520 $EJECT
```

APPENDIX D: 8298 E²PROM CONTROLLER FIRMWARE LISTING

```
LOC  OBJ        LINE        SOURCE STATEMENT
                  521
                  522
                  523       ;       ********************************************
                  524
                  525       ;               WRITE - WRITE TO 2816
                  526
                  527       ;               USES:   R0,A
                  528
                  529       ;       ********************************************
                  530
                  531 WRITE:
0145 3407         532       CALL    READ            ; READ DATA FIRST
0147 37           533       CPL     A               ; NEGATE DATA
0148 A8           534       MOV     R0,A            ; SAVE COMPLEMENTED DATA
0149 17           535       INC     A               ; NEGATE A
014A 6D           536       ADD     A,R5            ; A = (DATA TO WRITE - DATA READ)
014B C658         537       JZ      WREND           ; IF SAME DATA, DON'T BOTHER WRITING
014D F8           538       MOV     A,R0            ; RETRIEVE NEGATED DATA FROM READ CYCLE
014E 5D           539       ANL     A,R5            ; ANY 0'S TO BE PROGRAMMED TO 1'S?
014F C655         540       JZ      WRCYCL          ; NO -> SKIP ERASE BYTE ROUTINE
0151 23FF         541       MOV     A,#0FFH         ; SET DATA TO WRITE = 0FFH
0153 3459         542       CALL    WECYCL          ; ERASE THE BYTE
                  543 WRCYCL:
0155 FD           544       MOV     A,R5            ; WRITE DATA NOW
0156 3459         545       CALL    WECYCL
0158 83           546 WREND:  RET                 ; RETURN BACK TO CALLING PROGRAM
                  547
                  548
                  549       ;       ********************************************
                  550
                  551       ;               WECYCL - WRITE DATA IN A-REG TO 2816
                  552
                  553       ;               ASSUMES:        EE ENABLED, ADDRESSES OUTPUT
                  554
                  555       ;               USES:   R0
                  556
                  557       ;       ********************************************
                  558 WECYCL:
0159 39           559       OUTL    P1,A            ; OUTPUT DATA TO WRITE
015A 230E         560       MOV     A,#VPPACT       ; ACTIVATE VPP
015C 9F           561       ANLD    P7,A            ; ACTIVATE VPP
015D B81D         562       MOV     R0,#EESTAT      ; LET R0 POINT TO EE STATUS REGISTER
015F F0           563       MOV     A,@R0           ; GET EE STATUS
0160 537F         564       ANL     A,#INITWR       ; CLEAR WRITE COMPLETE BIT
0162 A0           565       MOV     @R0,A           ; RESTORE STATUS REGISTER
0163 D5           566       SEL     RB1             ; CHOOSE RB1 TO GET INITIAL TIMER VALUE
0164 85           567       CLR     F0              ; SET F0 TO INDICATE VPP IS ON
0165 95           568       CPL     F0
0166 FE           569       MOV     A,R6            ; START TIMER
0167 62           570       MOV     T,A
0168 55           571       STRT    T
0169 25           572       EN      TCNTI           ; ENABLE TIMER/COUNTER INTERRUPTS
                  573 WRWAIT:
016A 3483         574       CALL    CKDBB           ; CHECK FOR ABORT COMMAND IN DBB
016C FD           575       MOV     A,R5            ; GET EE STATUS
```

APPENDIX D: 8298 E^{2}PROM CONTROLLER FIRMWARE LISTING

```
LOC  OBJ        LINE       SOURCE STATEMENT

016D F271        576        JB7     WRENDC          ; SKIP RECHECK IF END OF CYCLE
016F 246A        577        JMP     WRWAIT
                 578 WRENDC:
0171 3474        579        CALL    SHUT            ; END OF WRITE CYCLE - SHUT DOWN 2816
0173 93          580        RETR                    ; RETURN BACK TO USER PROGRAM
                 581
                 582
                 583        ;       SHUT - SHUT OFF VPP, WAIT VPP FALL AND DEACTIVATE DATA LINES
                 584
                 585 SHUT:
0174 23F1        586        MOV     A,#VPPINA       ; DE-ACTIVATE VPP
0176 8F          587        ORLD    P7,A
0177 230E        588        MOV     A,#VPPFALL      ; WAIT WHILE VPP FALLS
0179 85          589        CLR     F0              ; CLEAR F0 TO INDICATE VPP IS OFF
017A 347F        590        CALL    DELAY
017C 89FF        591        ORL     P1,#0FFH        ; DEACTIVATE DATA LINES
017E 83          592        RET
                 593
                 594
                 595        ;       DELAY - SUBROUTINE TO DELAY
                 596
                 597        ;       PASS:   A-REG = COUNT (7.5 USEC/LOOP)
                 598
                 599 DELAY:
017F 07          600        DEC     A               ; DECREMENT A-REG
0180 967F        601        JNZ     DELAY           ; LOOP IF NOT ZERO
0182 83          602        RET                     ; BACK TO CALLER
                 603
                 604
                 605        ;       CKDBB - CHECK FOR ABORT COMMAND IN DBB
                 606
                 607        ;       USES:   A-REG
                 608
                 609 CKDBB:
0183 D687        610        JNIBF   CKDEX           ; IF INPUT BUFFER NOT FULL THEN EXIT
0185 7688        611        JF1     CKD10           ; IF COMMAND WAITING, CHECK IT
                 612 CKDEX:
0187 83          613        RET                     ; RETURN TO USER
                 614 CKD10:
0188 22          615        IN      A,DBB           ; GET COMMAND
0189 A5          616        CLR     F1              ; CLEAR COMMAND/DATA FLAG
018A 03F6        617        ADD     A,#-ABORTC      ; IS IT AN ABORT COMMAND?
018C 9690        618        JNZ     ILLCD1          , NO - ILLEGAL COMMAND
018E 6410        619        JMP     ABORT           ; GO ABORT
                 620 ILLCD1:
0190 23AB        621        MOV     A,#0ABH         ; OUTPUT ILLEGAL COMMAND MESSAGE
0192 02          622        OUT     DBB,A
0193 8A10        623        ORL     P2,#OBFACT      ; ALLOW OBF HOST INTERRRUPTS
0195 83          624        RET
                 625
                 626 $EJECT
```

APPENDIX D: 8298 E^{2}PROM CONTROLLER FIRMWARE LISTING

```
LOC  OBJ        LINE        SOURCE STATEMENT
                  627
                  628       ;     ********************************************
                  629
                  630       ;               GENERAL SUBROUTINES
                  631
                  632       ;     ********************************************
                  633
                  634
                  635
                  636       ;     INCADR - INCREMENT ADDRESS REGISTER
                  637
                  638       ;     USES:   RB0 - ANY REGISTER BANK
                  639       ;             A-REG
                  640
                  641 INCADR:
0196 B814         642         MOV   R0,#ADRLO       ; POINT TO LOW ADDRESS REGISTER
0198 10           643         INC   @R0             ; INCREMENT LOW ORDER ADDRESS
0199 F0           644         MOV   A,@R0           ; GET LOW ORDER ADDRESS
019A 969E         645         JNZ   INCEX           ; IF NONZERO, EXIT
019C 18           646         INC   R0              ; OVERFLOW - POINT TO HI ORDER ADDRESS REGISTER
019D 10           647         INC   @R0             ; INCREMENT HI ORDER ADDRESS
019E 83           648 INCEX:  RET                   ; RETURN TO USER PROGRAM
                  649
                  650
                  651
                  652       ;     DECCNT - DECREMENT COUNT REGISTER
                  653
                  654       ;     RETURNS:        A-REG = 0 IF COUNT = 0
                  655
                  656       ;     USES:           R0, A-REG
                  657
                  658 DECCNT:
019F B806         659         MOV   R0,#CNTLO       ; POINT TO LOW ORDER COUNT REGISTER
01A1 F0           660         MOV   A,@R0           ; GET COUNT
01A2 07           661         DEC   A               ; DECREMENT LOW ORDER COUNT
01A3 A0           662         MOV   @R0,A           ; RESTORE COUNT
01A4 C6AF         663         JZ    ZTEST           ; IF ALREADY ZERO, TEST FOR 0 COUNT
01A6 17           664         INC   A               ; SEE IF WE NEED TO BORROW
01A7 96AE         665         JNZ   DECEX           ; NO - JUST EXIT
01A9 18           666         INC   R0              ; YES - POINT TO HI ORDER COUNT REGISTER
01AA F0           667         MOV   A,@R0           ; GET HI ORDER COUNT VALUE
01AB 07           668         DEC   A               ; DECREMENT IT
01AC A0           669         MOV   @R0,A           ; RESTORE COUNT VALUE
01AD 17           670         INC   A               ; SET A = NONZERO TO INDICATE NOT 0 COUNT
01AE 83           671 DECEX:  RET                   ; RETURN TO USER
                  672 ZTEST:
01AF 18           673         INC   R0              ; POINT TO HI ORDER COUNT REG TO SEE IF IT'S 0
01B0 40           674         ORL   A,@R0           ; OR IN HI ORDER COUNT
01B1 24AE         675         JMP   DECEX           ; A = 0 IF HI OR LOW = 0 SO EXIT
                  676
                  677
                  678 $EJECT
```

APPENDIX D: 8298 E²PROM CONTROLLER FIRMWARE LISTING

```
LOC  OBJ        LINE        SOURCE STATEMENT

                679
                680         ;       ************************************************
                681
                682         ;                   COMMAND   DRIVER
                683
                684         ;       ************************************************
                685
                686
0200            687         ORG     200H            ; START COMMANDS ON PAGE 2
                688
                689 FIRSTI:
                690
                691         ;       ILLEGAL COMMAND
                692
                693 IC:
0200 042E       694         JMP     ILLCMD          ; ILLEGAL COMMAND
                695
                696
                697         ;       READ COMMAND
                698
                699 READC:
0202 3428       700         CALL    REQACC          ; REQUEST ACCESS TO LOCAL BUS
0204 3407       701         CALL    READ            ; READ FROM 2816
                702 RDCPL:
0206 540A       703         CALL    OUTPUT          ; OUTPUT DATA TO DBB
0208 0423       704 CMDEX:  JMP     CMDCPL          ; COMMAND EXITS HERE
                705
                706
                707         ;       OUTPUT - OUTPUT DATA IN A-REG TO DBB
                708
                709 OUTPUT:
020A 8A10       710         ORL     P2,#OBFACT      ; ACTIVATE OBF INTERRUPT
020C 02         711         OUT     DBB,A           ; OUTPUT DATA TO HOST
020D 860D       712 WAITOU: JOBF    WAITOU          ; WAIT UNTIL OUTPUTPUT BUFR NOT FULL
020F 83         713         RET
                714
                715
                716         ;       WRITE COMMAND
                717
                718 WRITEC:
0210 3428       719         CALL    REQACC          ; REQUEST ACCESS TO LOCAL BUS
0212 3445       720         CALL    WRITE           ; WRITE TO 2816
0214 0423       721         JMP     CMDCPL          ; COMMAND COMPLETE
                722
                723
                724         ;       CHIP ERASE COMMAND
                725
                726 CERASE:
0216 3428       727         CALL    REQACC          ; REQUEST ACCESS TO BUS
0218 B815       728         MOV     R0,#ADRHI       ; POINT TO HI ORDER ADDRESS
                729
                730         ;       OUTPUT A12,A13 TO P26,P27
                731
021A F0         732         MOV     A,@R0           ; GET HI ORDER ADDRESS
021B E7         733         RL      A               ; MOVE TO BITS 6,7
```

APPENDIX D: 8298 E²PROM CONTROLLER FIRMWARE LISTING

```
LOC  OBJ        LINE       SOURCE STATEMENT

021C E7          734        RL      A
021D 53C0        735        ANL     A,#0C0H         ; MAKE SURE BITS 6,7 THE ONLY ONES SET
021F 3A          736        OUTL    P2,A            ; OUTPUT HI ORDER ADDRESS BITS
0220 F0          737        MOV     A,@R0           ; GET HI ORDER ADDRESS
0221 997F        738        ANL     P1,#7FH         ; SET D7/CERASE' = 0 (DATA LINE)
0223 230B        739        MOV     A,#EENACT       ; ACTIVATE EEN' TO LATCH CERASE' BIT
0225 9F          740        ANLD    P7,A
0226 F0          741        MOV     A,@R0           ; GET HI ORDER ADDRESS AGAIN
0227 3E          742        MOVD    P6,A            ; OUTPUT ADDRESS BITS A8-A11
0228 23FF        743        MOV     A,#0FFH         ; SET DATA TO WRITE = 0FFH FOR CHIP ERASE
022A 3459        744        CALL    WECYCL          ; ERASE THE SELECTED CHIP
022C 5430        745        CALL    REMCE           ; REMOVE CHIP ERASE
022E 0423        746        JMP     CMDCPL          ; END OF COMMAND
                 747
                 748
                 749        ;       REMCE - REMOVE CHIP ERASE (OE'=12V)
                 750
                 751        ;       USES A-REG
                 752
                 753 REMCE:
0230 89FF        754        ORL     P1,#0FFH        ; SET DATA LINES = 0FFH
0232 2304        755        MOV     A,#EENINA       ; DE-ACTIVATE EEN
0234 8F          756        ORLD    P7,A
0235 230B        757        MOV     A,#EENACT       ; D7/CERASE' = 1 (SINCE DATA WE WRITE = FFH)
                 758                                ; JUST PULSE EEN TO LATCH INACTIVE CERASE'
0237 9F          759        ANLD    P7,A            ; PULSE EEN' TO LATCH INACTIVE CERASE'
0238 83          760        RET
                 761
                 762
                 763        ;       BLOCK ERASE
                 764
                 765 BLOCKE:
0239 3428        766        CALL    REQACC          ; REQUEST ACCESS TO BUS
023B BDFF        767        MOV     R5,#0FFH        ; SE DATA TO WRITE = 0FFH
                 768 BL10:
023D 3445        769        CALL    WRITE           ; WRITE ERASED BYTE
023F 343A        770        CALL    REMADR          ; DE-ACTIVATE ADDRESS LINES
0241 3496        771        CALL    INCADR          ; INCREMENT EE ADDRESS
0243 EE3D        772        DJNZ    R6,BL10         ; KEEP LOOPING IF NOT DONE
0245 0423        773        JMP     CMDCPL          ; WHEN DONE EXIT
                 774
                 775
                 776        ;       MULTIPLE WRITE
                 777
                 778 MULTWR:
0247 BC00        779        MOV     R4,#0           ; ZERO BUFFER COUNT
0249 D5          780        SEL     RB1             ; SELECT ALTERNATE REGISTER SET
024A BF20        781        MOV     R7,#BUFST       ; POINT TO START OF BUFFER
                 782 MWGET:
024C D5          783        SEL     RB1             ; MAKE SURE WE'RE POINTING TO RB1
024D FB          784        MOV     A,R3            ; GET DBB STATUS
024E 53E0        785        ANL     A,#0E0H         ; ZERO TYPE & BYTE STATUS
0250 431D        786        ORL     A,#11101B       ; SET TYPE=7, 3 BYTES TO GET (SPECIAL TYPE)
0252 AB          787        MOV     R3,A            ; RESTORE DBB STATUS
0253 149F        788        CALL    GETDAT          ; GET 3 BYTES FROM HOST
```

APPENDIX D: 8298 E²PROM CONTROLLER FIRMWARE LISTING

```
LOC  OBJ        LINE        SOURCE STATEMENT
0255 FF          789          MOV     A,R7            ; GET DATA DESTINATION POINTER
0256 C5          790          SEL     RB0
0257 1C          791          INC     R4              ; BUMP BUFFER COUNTER
0258 03C2        792          ADD     A,#(-MWBEND)    ; CHECK FOR END OF BUFFER
025A C65F        793          JZ      MWDUMP          ; IF END GO DUMP BUFFER
025C EE4C        794          DJNZ    R6,MWGET        ; IF NOT DONE THEN GET MORE
025E 1E          795          INC     R6              ; INCREMENT BECAUSE WE'LL DEC NEXT CYCLE
                 796 MWDUMP:
025F CE          797          DEC     R6              ; DECREMENT BECAUSE MAY NOT HAVE DONE SO
0260 B920        798          MOV     R1,#BUFST       ; POINT TO START OF BUFFER
                 799 MWDMP1:
0262 B815        800          MOV     R0,#ADRHI       ; POINT TO HI ORDER ADDRESS
0264 3428        801          CALL    REQACC          ; REQUEST ACCESS TO BUS
0266 F1          802          MOV     A,@R1           ; GET HI ADDRESS AGAIN
0267 A0          803          MOV     @R0,A           ; SAVE IN HI ADDRESS REGISTER
0268 C8          804          DEC     R0              ; POINT TO LOW ADDRESS REG
0269 19          805          INC     R1              ; BUMP BUFFER POINTER
026A F1          806          MOV     A,@R1           ; GET LOW ORDER ADDRESS
026B A0          807          MOV     @R0,A           ; AND SAVE IN LOW ORDER ADDRESS REGISTER
026C 18          808          INC     R0              ; POINT TO HI ADDRESS
026D 19          809          INC     R1              ; BUMP BUFFER POINTER
026E F1          810          MOV     A,@R1           ; GET DATA TO WRITE
026F 19          811          INC     R1              ; BUMP BUFFER POINTER
0270 AD          812          MOV     R5,A            ; SAVE IN DATA WRITE REGISTER
0271 3445        813          CALL    WRITE           ; WRITE TO 2816
0273 343A        814          CALL    REMADR          ; DE-ACTIVATE ADDRESS LINES
0275 EC62        815          DJNZ    R4,MWDMP1       ; DECREMENT COUNT - KEEP WRITING IF NOT DONE
0277 FE          816          MOV     A,R6            ; PICK UP COUNT TO SEE IF WE'VE DONE ALL BYTES
0278 C67C        817          JZ      MWEX            ; IF DONE THEN EXIT
027A 4447        818          JMP     MULTWR          ; IF NOT DONE THEN GET ANOTHER BLOCK
027C 0423        819 MWEX:    JMP     CMDCPL          ; EXIT TO COMMAND COMPLETE ROUTINE
                 820
                 821
                 822          ;       SERIES WRITE
                 823
                 824 SERWRT:
027E BC00        825          MOV     R4,#0           ; ZERO BUFFER COUNTER
0280 D5          826          SEL     RB1             ; POINT TO ALTERNATE REGISTER
0281 BF20        827          MOV     R7,#BUFST       ; POINT TO START OF BUFFER IN DESTINATION REG
                 828 SWGET:
0283 D5          829          SEL     RB1             ; MAKE SURE IN REGISTER BANK 1
0284 FB          830          MOV     A,R3            ; GET DBB STATUS
0285 53E0        831          ANL     A,#0E0H         ; REMOVE TYPE AND BYTE BITS
0287 431F        832          ORL     A,#11111B       ; OR INT TYPE=4, 1 TYPE TO GET
0289 AB          833          MOV     R3,A            ; RESTORE DBB STATUS
028A 149F        834          CALL    GETDAT          ; GET 1 BYTE FROM HOST
028C FF          835          MOV     A,R7            ; GET BUFFER POINTER
028D C5          836          SEL     RB0             ; POINT TO NORMAL REGISTER SET
028E 1C          837          INC     R4              ; INCREMENT BUFFER COUNTER
028F 03C0        838          ADD     A,#-SWBEND      ; CHECK FOR END OF BUFFER
0291 C699        839          JZ      SWDMP0          ; IF END OF BUFFER THEN GO DUMP DATA TO EEPROM
0293 349F        840          CALL    DECCNT          ; DECREMENT COUNT
0295 9683        841          JNZ     SWGET           ; IF COUNT NOT ZERO THEN GET ANOTHER BYTE
0297 449B        842          JMP     SWDUMP          ; SKIP OVER FIX, SINCE K-CODE MAKES IT NEC.
                 843 SWDMP0:
```

APPENDIX D: 8298 E²PROM CONTROLLER FIRMWARE LISTING

```
LOC  OBJ        LINE       SOURCE STATEMENT
0299 349F        844        CALL    DECCNT          ; SKIPPED DECREMENT ABOVE, SO FIX NOW
                 845 SWDUMP:
029B B920        846        MOV     R1,#BUFST       ; POINT TO START OF BUFFER
029D 3428        847        CALL    REQACC          ; REQUEST ACCESS TO INTERNAL BUSSES
                 848 SWDMP1:
029F F1          849        MOV     A,@R1           ; GET A BYTE OF DATA
02A0 AD          850        MOV     R5,A            ; SAVE DATA TO WRITE
02A1 3445        851        CALL    WRITE           ; WRITE TO 2816
02A3 343A        852        CALL    REMADR          ; RELEASE ADDRESS LINES
02A5 19          853        INC     R1              ; BUMP POINTER
02A6 3496        854        CALL    INCADR          ; INCREMENT EE ADDRESS
02A8 EC9F        855        DJNZ    R4,SWDMP1       ; DECREMENT COUNT - IF NOT ZERO, WRITE NXT BYTE
                 856
                 857        ;       DONE WRITING BUFFER - CHECK IF DONE WITH ALL BYTES
                 858
02AA FE          859        MOV     A,R6            ; CHECK IF COUNT BYTES ARE 0
02AB 4F          860        ORL     A,R7            ; OR IN HI AND LOW ORDER COUNT BYTES
02AC 967E        861        JNZ     SERWRT          ; IF NONZERO, GET ANOTHER BUFFER FULL OF DATA
02AE 0423        862        JMP     CMDCPL          ; IF DONE GO TO COMMAND COMPLETE ROUTINE
                 863
                 864
                 865        ;       SERIES READ COMMAND
                 866
                 867 SERRD:
02B0 3428        868        CALL    REQACC          ; REQUEST ACCESS TO LOCAL BUSSES
02B2 3407        869        CALL    READ            ; READ FROM 2816 MEMORY
02B4 540A        870        CALL    OUTPUT          ; OUTPUT SERIALLY OR TO DATA BUS BUFFER
02B6 343A        871        CALL    REMADR          ; RELEASE ADDRESS PINS
02B8 3496        872        CALL    INCADR          ; INCREMENT EE ADDRESS
02BA 349F        873        CALL    DECCNT          ; DECREMENT COUNT AND SET Z FLAG IF COUNT=0
02BC 96B0        874        JNZ     SERRD           ; READ AGAIN IF NOT DONE
02BE 0423        875        JMP     CMDCPL          ; GO TO COMMAND COMPLETE ROUTINE
                 876
                 877
                 878
02C0 4400        879 ICDRD:  JMP     IC              ; OFF-PAGE REFERENCE TO ILLEGAL COMMAND ROUTINE
                 880
                 881
0300             882        ORG     300H
                 883
                 884 FIRSTP3:
                 885
                 886        ;       ENABLE DIRECT WRITE COMMAND
                 887
                 888 ENDW:
0300 D5          889        SEL     RB1             ; SELECT ALTERNATE REGISTER SET
0301 FB          890        MOV     A,R3            ; GET DBB STATUS
0302 4340        891        ORL     A,#040H         ; SET DIRECT WRITE BIT
0304 AB          892        MOV     R3,A            ; RESTORE DBB STATUS
0305 85          893        CLR     F0              ; SET DIRECT WRITE POSSIBLE FLAG
0306 95          894        CPL     F0
0307 0423        895        JMP     CMDCPL          ; COMMAND COMPLETE
                 896
                 897
                 898        ;       DISABLE DIRECT WRITE COMMAND
```

APPENDIX D: 8298 E^2PROM CONTROLLER FIRMWARE LISTING

```
LOC  OBJ        LINE      SOURCE STATEMENT
                  899
                  900 DISDW:
0309 D5           901         SEL     RB1             ; POINT TO ALTERNATE REGISTER SET
030A FB           902         MOV     A,R3            ; GET DBB STATUS
030B 53BF         903         ANL     A,#NOT 40H      ; CLEAR D/W BIT AS WELL AS EXTRANEOUS BIT
030D AB           904         MOV     R3,A            ; RESTORE DBB STATUS
030E 0423         905         JMP     CMDCPL          ; GO TO COMMAND COMPLETE ROUTINE
                  906
                  907
                  908         ;       ABORT - DO A SOFTWARE RESET
                  909
                  910 ABORT:
0310 3F           911         MOVD    P7,A            ; SEE IF VPP ON
0311 1215         912         JB0     AB10            ; IF YES, TURN OFF VPP
0313 3474         913         CALL    SHUT
                  914 AB10:
0315 5430         915         CALL    REMCE           ; REMOVE CHIP ERASE IN CASE IT WAS ON
0317 0423         916         JMP     CMDCPL          ; END OF COMMAND
                  917
                  918
                  919         ;       COMMANDS TO BE USED AFTER AN ABORT
                  920
                  921         ;       READ LOW ADDRESS COMMAND
                  922
                  923 READAL:
0319 B814         924         MOV     R0,#ADRLO       ; POINT TO LOW ORDER ADDRESS REG.
031B F0           925         MOV     A,@R0           ; READ IT INTO ACC.
031C 07           926         DEC     A               ; DECREMENT TO GIVE CORRECT VALUE
031D 4406         927         JMP     RDCPL
                  928
                  929         ;       READ HIGH ADDRESS COMMAND
                  930
                  931 READAH:
031F B815         932         MOV     R0,#ADRHI       ; POINT TO HIGH ORDER ADDRESS REG.
0321 F0           933         MOV     A,@R0
0322 4406         934         JMP     RDCPL
                  935
                  936         ;       READ WRITE DATA COMMAND
                  937
                  938 READWR:
0324 FD           939         MOV     A,R5
0325 4406         940         JMP     RDCPL
                  941
                  942 $EJECT
```

APPENDIX D: 8298 E²PROM CONTROLLER FIRMWARE LISTING

```
LOC  OBJ        LINE      SOURCE STATEMENT
                  943
                  944       ;       ************************************************
                  945
                  946       ;               PAGE 3 DATA TABLES
                  947
                  948       ;       ************************************************
                  949
0365              950       ORG     365H
                  951
                  952
                  953
                  954       ;       INSTRUCTION TABLE
                  955
                  956       ;       FORMAT:
                  957       ;
                  958       ;       1ST BYTE = COMMAND DESCRIPTOR:
                  959       ;
                  960       ;               BIT(S)          DESCRIPTION
                  961       ;               ------          -----------
                  962       ;               0-1             STARTING BYTE # (0-3) OF DATA BYTES
                  963       ;                               TO GET FOR THIS COMMAND
                  964       ;               2-4             TYPE (FROM DATA DEST TABLE) OF DATA
                  965       ;                               BYTES TO GET
                  966       ;                5              NO DATA BYTES TO RECEIVE FOR THIS
                  967       ;                               COMMAND (=1)
                  968       ;                6              PAGE 3 COMMAND
                  969       ;                7              COMMAND CAN BE EXECUTED UNDER SERIAL
                  970       ;                               I/O MODE
                  971       ;
                  972       ;       2ND BYTE = LOW ORDER STARTING ADDRESS OF COMMAND
                  973
                  974       ;       COMMENT FORMAT: CMND #,COMMAND, TYPE OF DATA, # BYTES TO GET
                  975
                  976
                  977 INSTBL:
0365 20           978       DB      20H,LOW IC              ; SUB COMMAND - WE SHOULDN'T GET HERE
0366 00
0367 83           979       DB      10000011B,LOW READC     ; 1 = READ, TYPE=0, BYTE=1
0368 02
0369 86           980       DB      10000110B, LOW WRITEC   ; 2 = WRITE, TYPE=1,BYTES=2
036A 10
036B 20           981       DB      20H,LOW IC              ; 3 = IC
036C 00
                  982
                  983       ;       SUB COMMANDS
                  984
                  985 SUBTBL:
036D 20           986       DB      20H,LOW IC              ; 0 = ILLEGAL COMMAND
036E 00
036F 20           987       DB      20H,LOW IC              ; 1 = ILLEGAL COMMAND
0370 00
0371 20           988       DB      20H,LOW IC              ; 2 = IC
0372 00
0373 20           989       DB      20H,LOW IC              ; 3 = IC
0374 00
```

APPENDIX D: 8298 E^{2}PROM CONTROLLER FIRMWARE LISTING

```
LOC  OBJ        LINE      SOURCE STATEMENT

0375 20          990       DB      20H,LOW IC              ; 4 = IC
0376 00
0377 09          991       DB      00001001B,LOW BLOCKE    ; 5 = BLOCK ERASE, TYPE=2, BYTES=3
0378 39
0379 9B          992       DB      10011011B,LOW CERASE    ; 6 = CHIP ERASE, TYPE=6, BYTES=1
037A 16
037B 17          993       DB      00010111B,LOW CMDEX     ; 7 = INIT WRITE TIME, TYPE=5,BYTES=1
037C 08
037D 20          994       DB      20H,LOW IC              ; 8 = IC
037E 00
037F 20          995       DB      20H,LOW IC              ; 9 = IC
0380 00
0381 60          996       DB      01100000B,LOW ABORT     ; A = ABORT, TYPE=0, BYTES=0
0382 10
0383 20          997       DB      20H,LOW IC              ; B = IC
0384 00
0385 8B          998       DB      10001011B,LOW MULTWR    ; C = MULTIPLE WRITE, TYPE=2,BYTE=1
0386 47
                 999
                 1000      REPT    19D
                 1001      DB      20H,LOW IC              ; D-1F = ILLEGAL COMMAND
                 1002      ENDM
0387 20          1003+     DB      20H,LOW IC              ; D-1F = ILLEGAL COMMAND
0388 00
0389 20          1004+     DB      20H,LOW IC              ; D-1F = ILLEGAL COMMAND
038A 00
038B 20          1005+     DB      20H,LOW IC              ; D-1F = ILLEGAL COMMAND
038C 00
038D 20          1006+     DB      20H,LOW IC              ; D-1F = ILLEGAL COMMAND
038E 00
038F 20          1007+     DB      20H,LOW IC              ; D-1F = ILLEGAL COMMAND
0390 00
0391 20          1008+     DB      20H,LOW IC              ; D-1F = ILLEGAL COMMAND
0392 00
0393 20          1009+     DB      20H,LOW IC              ; D-1F = ILLEGAL COMMAND
0394 00
0395 20          1010+     DB      20H,LOW IC              ; D-1F = ILLEGAL COMMAND
0396 00
0397 20          1011+     DB      20H,LOW IC              ; D-1F = ILLEGAL COMMAND
0398 00
0399 20          1012+     DB      20H,LOW IC              ; D-1F = ILLEGAL COMMAND
039A 00
039B 20          1013+     DB      20H,LOW IC              ; D-1F = ILLEGAL COMMAND
039C 00
039D 20          1014+     DB      20H,LOW IC              ; D-1F = ILLEGAL COMMAND
039E 00
039F 20          1015+     DB      20H,LOW IC              ; D-1F = ILLEGAL COMMAND
03A0 00
03A1 20          1016+     DB      20H,LOW IC              ; D-1F = ILLEGAL COMMAND
03A2 00
03A3 20          1017+     DB      20H,LOW IC              ; D-1F = ILLEGAL COMMAND
03A4 00
03A5 20          1018+     DB      20H,LOW IC              ; D-1F = ILLEGAL COMMAND
03A6 00
03A7 20          1019+     DB      20H,LOW IC              ; D-1F = ILLEGAL COMMAND
```

APPENDIX D: 8298 E²PROM CONTROLLER FIRMWARE LISTING

```
LOC  OBJ        LINE       SOURCE STATEMENT
03A8 00
03A9 20         1020+       DB      20H,LOW IC              ; D-1F = ILLEGAL COMMAND
03AA 00
03AB 20         1021+       DB      20H,LOW IC              ; D-1F = ILLEGAL COMMAND
03AC 00
                1022
03AD 20         1023        DB      20H,LOW IC              ; 20 = IC
03AE 00
03AF 20         1024        DB      20H,LOW IC              ; 21 = IC
03B0 00
03B1 60         1025        DB      01100000B,LOW ENDW      ; 22 = ENABLE D/W
03B2 00
03B3 60         1026        DB      01100000B,LOW DISDW     ; 23 = DISABLE D/W
03B4 09
03B5 8C         1027        DB      10001100B,LOW SERRD     ; 24 = SERIES READ, TYPE=3, BYTES=4
03B6 B0
03B7 8C         1028        DB      10001100B,LOW SERWRT    ; 25 = SERIES WRITE,TYPE=3, BYTES=4
03B8 7E
03B9 20         1029        DB      20H,LOW IC              ; 26 = IC
03BA 00
03BB 20         1030        DB      20H,LOW IC              ; 27 = IC
03BC 00
03BD 60         1031        DB      01100000B,LOW READAL    ; 28 = READ LOW ADDR, NO DATA
03BE 19
03BF 60         1032        DB      01100000B,LOW READAH    ; 29 = READ HI  ADDR, NO DATA
03C0 1F
03C1 60         1033        DB      01100000B,LOW READWR    ; 2A = READ WRITE DATA, NO DATA TO RCV
03C2 24
                1034
                1035        REPT 5D
                1036        DB      20H,LOW IC              ; 2B-3F = ILLEGAL COMMAND
                1037        ENDM                            ; 30-3F DETECTED EARLIER
03C3 20         1038+       DB      20H,LOW IC              ; 2B-3F = ILLEGAL COMMAND
03C4 00
03C5 20         1039+       DB      20H,LOW IC              ; 2B-3F = ILLEGAL COMMAND
03C6 00
03C7 20         1040+       DB      20H,LOW IC              ; 2B-3F = ILLEGAL COMMAND
03C8 00
03C9 20         1041+       DB      20H,LOW IC              ; 2B-3F = ILLEGAL COMMAND
03CA 00
03CB 20         1042+       DB      20H,LOW IC              ; 2B-3F = ILLEGAL COMMAND
03CC 00
                1043
                1044
                1045        ;       DATA DESTINATION TABLE
                1046
                1047        ;       FORMAT:
                1048        ;               DB      ADR OF BYTE 3, BYTE 2, BYTE 1, BYTE 0 ;TYPE=N
                1049
                1050 DESTBL:
03CD 00         1051        DB      0,0,0,ADRLO             ; TYPE 0
03CE 00
03CF 00
03D0 14
03D1 00         1052        DB      0,ADRHI,ADRLO,WRTDAT    ; TYPE 1
```

APPENDIX D: 8298 E^2PROM CONTROLLER FIRMWARE LISTING

```
LOC  OBJ        LINE       SOURCE STATEMENT

03D2 15
03D3 14
03D4 05
03D5 00         1053       DB      0,ADRHI,ADRLO,CNTLO     ; TYPE 2
03D6 15
03D7 14
03D8 06
03D9 15         1054       DB      ADRHI,ADRLO,CNTHI,CNTLO ; TYPE 3
03DA 14
03DB 07
03DC 06
03DD 00         1055       DB      0,0,0,ADRLO             ; TYPE 4
03DE 00
03DF 00
03E0 14
03E1 00         1056       DB      0,0,0,INTIME            ; TYPE 5
03E2 00
03E3 00
03E4 1E
03E5 00         1057       DB      0,0,0,ADRHI             ; TYPE 6
03E6 00
03E7 00
03E8 15
                1058
                1059       END

USER SYMBOLS
ABIO   0315  ABORT  0310  ABORTC 000A  ADRHI  0015  ADRLO  0014  ASAVE  001C  BL10   023D  BLOCKE 0239
BUFCNT 0004  BUFST  0020  CERASE 0216  CKD10  0188  CKDBB  0183  CKDEX  0187  CMDCPL 0023  CMDEX  0208
CNTHI  0007  CNTLO  0006  COMFB  001A  DATDES 001F  DATRCV 00C7  DBBI   0033  DBBI07 003F  DBSTAT 001B
DECCNT 019F  DECEX  01AE  DELAY  017F  DESTBL 03CD  DISDW  0309  DW05   00E1  DW10   00F7  DWCMD  00DD
DWE    001F  DWNE   0020  EENACT 000B  EENINA 0004  EESTAT 001D  ENDW   0300  EXEC   00C3  EXIT   0072
FIRSTI 0200  FIRSTP 0300  GETCMD 0015  GETDAT 009F  GETDBB 00B3  GOTACC 012F  IBFACT FF20  IBFINA 00DF
IC     0200  ICDRD  02C0  ILLCD1 0190  ILLCDO 006D  ILLCMD 002E  INCADR 0196  INCEX  019E  ININT  0016
INIT   0009  INITWR 007F  INSTBL 0365  INTCNT 0003  INTIME 001E  LOOKUP 0056  MAYBIC 004B  MULTWR 0247
MWBEND 003E  MWDMP1 0262  MWDUMP 025F  MWEX   027C  MWGET  024C  NOTICD 004D  OBFACT 0010  OBFINA FFEF
OUTA20 011E  OUTADR 0113  OUTPUT 020A  P3INST 0098  PC03   0080  PC05   008D  PC10   009A  PCMND  007D
RDACT  0009  RDCPL  0206  RDINA  0002  READ   0107  READAH 031F  READAL 0319  READC  0202  READWR 0324
RELEAS 0130  RELRET 0138  REMADR 013A  REMCE  0230  REMRET 0141  REQACC 0128  REQACT 0007  RESET  0000
SAVEHI 00B9  SCR0   0000  SCR0P  0018  SCR1   0001  SCR1P  0019  SCR2   0002  SERRD  02B0  SERWRT 027E
SETGET 00A1  SHUT   0174  SUBCMD 0075  SUBTBL 036D  SWBEND 0040  SWDMP0 0299  SWDMP1 029F  SWDUMP 029B
SWGET  0283  TIME   FF83  TIMER  00FC  VPPACT 000E  VPPFAL 000E  VPPINA FFF1  WAITAC 012D  WAITB  00B5
WAITOU 020D  WECYCL 0159  WIPSTS 0007  WRCYCL 0155  WREND  0158  WRENDC 0171  WRITE  0145  WRITEC 0210
XRTDAT 0005  WRWAIT 016A  ZTEST  01AF
ASSEMBLY COMPLETE,  NO ERRORS
```

September 1981

A 2716 to 2816 Programming Socket Adapter

Tom Haverstock and Bill Carney
Special Products Division
Applications Engineering

©Intel Corporation, 1981.

INTRODUCTION

This application note will examine and discuss a socket adapter that allows the user to interface a 2816 E^2PROM with a 2716 EPROM programmer. The adapter permits the programmer to exercise the features of the 2816—read, byte write and chip erase. Compatability with most 2716 programmers is achieved through a small component count, thereby providing a cost-efficient means of programming E^2PROMS.

HARDWARE

The E^2 pinout, shown in Figure 1, is nearly identical to that of the industry standard, JEDEC approved 2716 EPROM. There are, however, several major differences in the three control signals, V_{PP}, CE, and OE. These signals require special circuitry to permit the 2816 to be programmed by a 2716 programmer. One of these circuits is for generating the 2816 programming voltage waveform, V_{PP}. Figure 2 exhibits the difference between the V_{PP} waveforms of the 2716 and

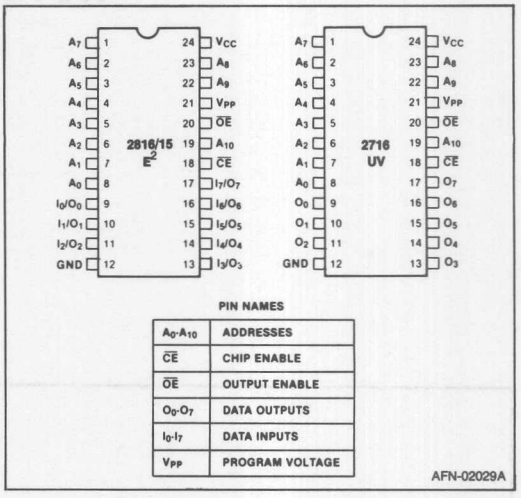


Figure 1. 2816 Pinout

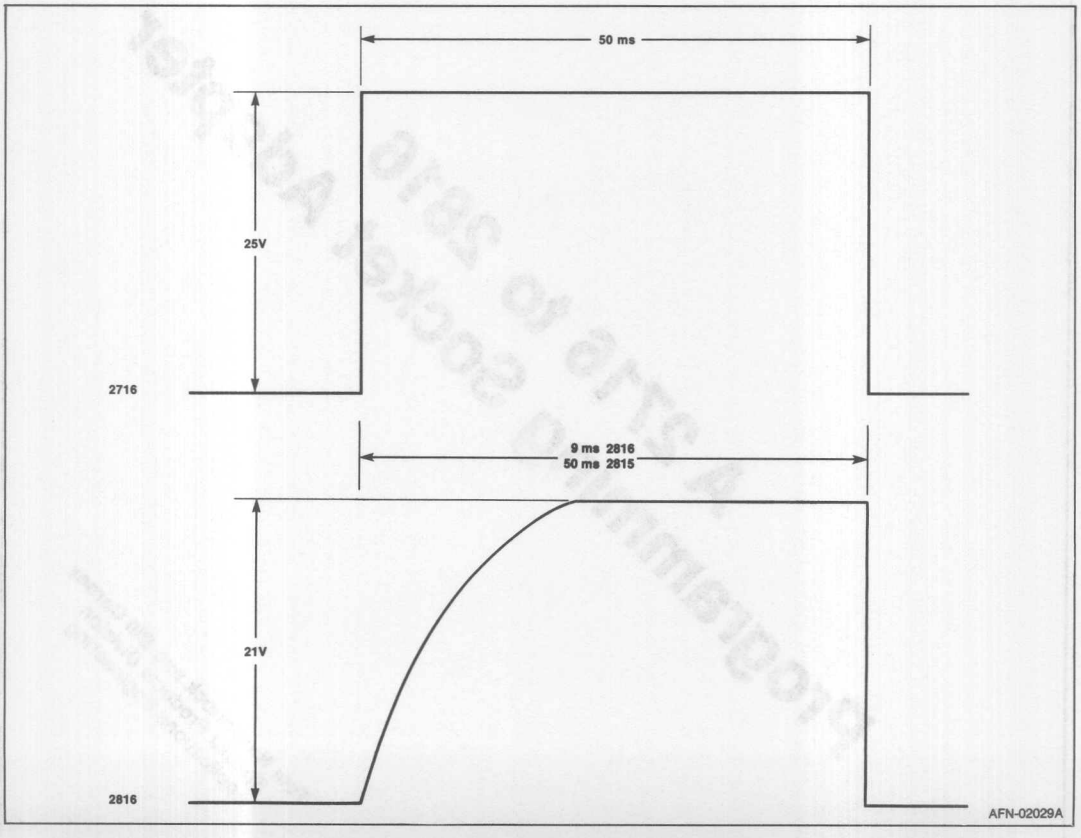


Figure 2. V_{PP} Waveform for 2716 and 2816

2816. The 2716 EPROM is programmed by a 25V, 50 ms long pulse per byte, whereas the 2816 E^2PROM requires a 21V pulse of 9 to 15 ms in duration. The 2815, a new 16K E^2PROM, requires a 50 ms pulse. The socket adapter can be set for either value, by a simple resistor/capacitor change. For the capability to program both, the pulse width can be set at 50 ms. Additionally, the V_{PP} pulse must possess a specific exponential rise time constant. Therefore, special waveshaping and timing circuitry is necessary to produce an E^2PROM programming pulse from the 2716 programmer's output. (The nature of the V_{PP} pulse is explained in detail in AP-101.)

Figure 3 shows the circuit that will produce the 2816/15 compatible programming pulse as well as the $\overline{CE}$ signal. The 4011 CMOS NAND gates are required to isolate the adapter from the programmer outputs. A TTL 7400 bipolar NAND gate cannot be used as its outputs would appear to the programmer as a leaky MOS device and can cause the programming operation to cease in some manufacturer's programmers.

Upon assertion of $\overline{CE}$ by the 2716 programmer, the adapter circuitry provides a $\overline{CE}$-out signal to the socket containing the 2816. If V_{PP} should be applied at this time, the 9602 one-shot is triggered and generates a pulse of approximately 12 ms. This pulse is then shaped by the dual op-amp circuit to produce the correct exponential rise necessary to program the 2816. The rise time constant is determined by the 10K resistor and the 0.05 μF capacitor and is nominally 500 μsec. The net result of the circuitry of Figure 3 is the transformation of a 25V, 50 ms square pulse into a 12 ms long, 21V programming pulse with a 500 μs rise time constant, and the generation of the CE signal. For the 2815, this pulse width is 50 ms.

In Figure 4, V_{PX} will pass uninterrupted to the 2816 through the DH0006 current driver when a programming operation is performed. If, however, the E^2PROM is to be chip erased, then the switch is placed as shown. This causes the digital feedback network, consisting of the two 74LS74 flip-flops, to be connected to the DH0006 current driver. This feedback has the

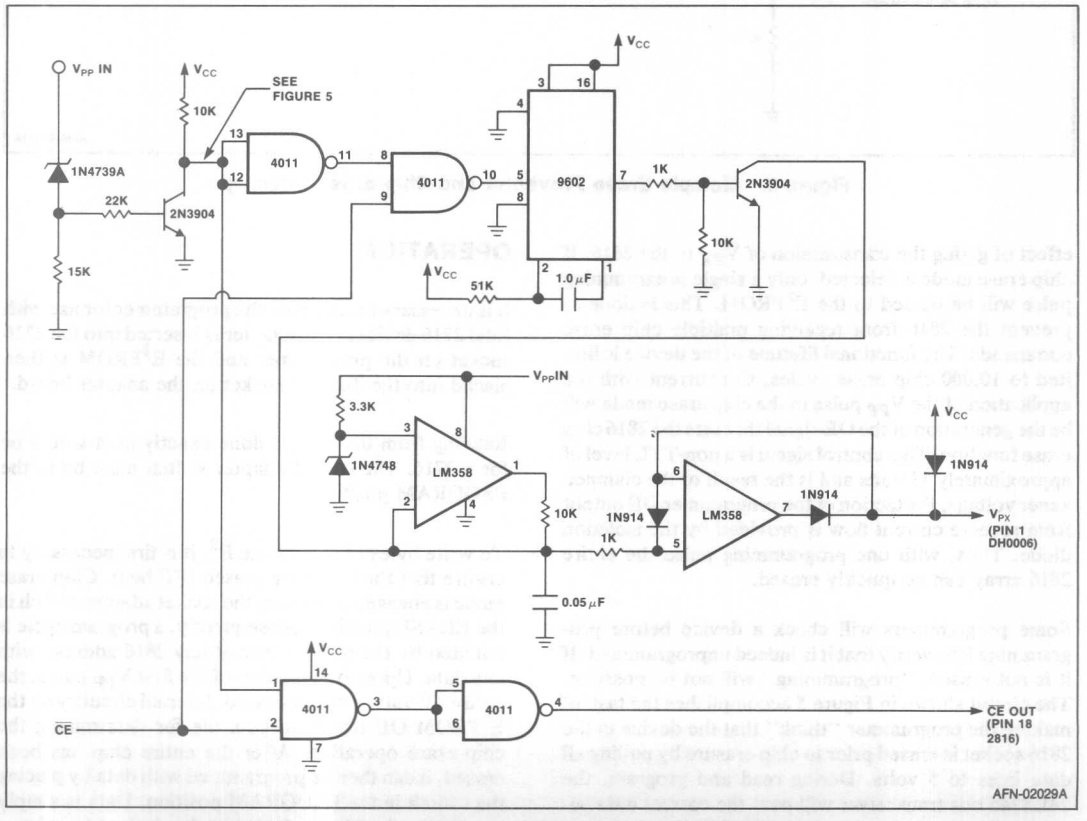


Figure 3. 2816 Socket Adapter for 2716 Programmers

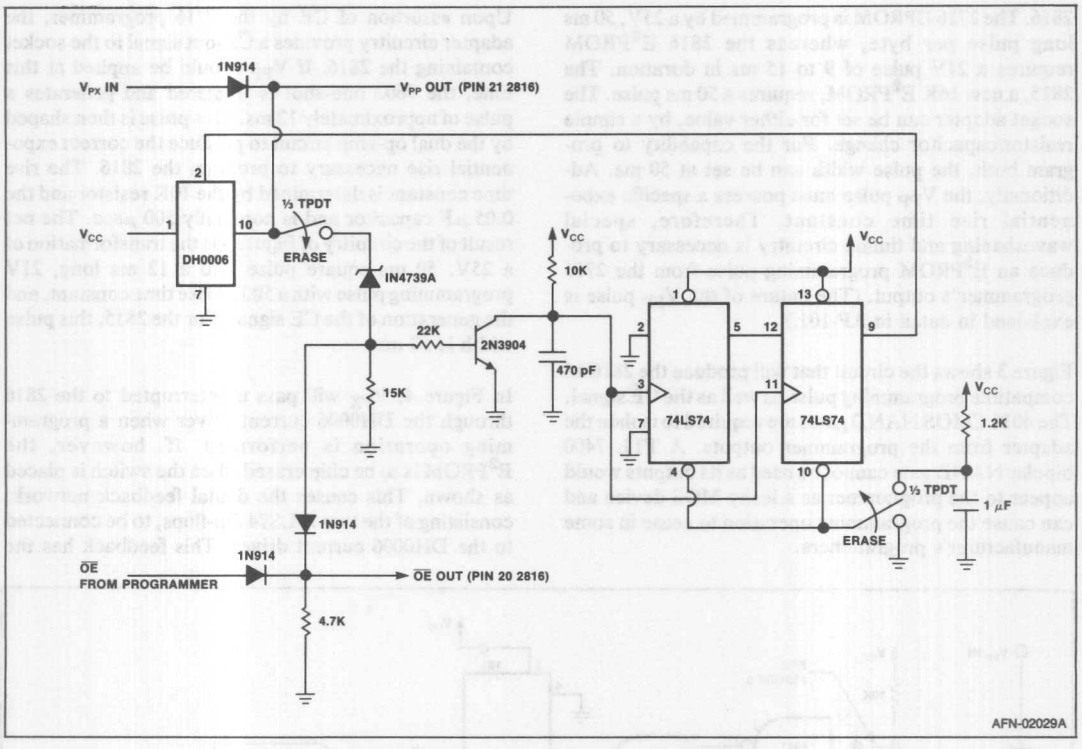


Figure 4. Multiple Erase Preventer and Chip Erase Circuitry

effect of gating the transmission of V_{PX} to the 2816. If chip erase mode is selected, only a single programming pulse will be passed to the E^2PROM. This is done to prevent the 2816 from receiving multiple chip erase commands. The functional lifetime of the device is limited to 10,000 chip erase cycles. Concurrent with the application of the V_{PP} pulse in the chip erase mode will be the generation of the OE sjgnal that sets the 2816 chip erase function. This control signal is a non-TTL level of approximately 13 volts and is the result of the clamped zener voltage. Protection of the programmer OE output from reverse current flow is provided by the isolation diode. Thus, with one programming pulse the entire 2816 array can be quickly erased.

Some programmers will check a device before programming it to verify that it is indeed unprogrammed. If it is not erased, "programming" will not be possible. The circuit shown in Figure 5 accomplishes the task of making the programmer "think" that the device in the 2816 socket is erased prior to chip erasure by pulling all data lines to 5 volts. During read and program, the 74LS245 bus transceiver will pass the correct data between the programmer and the E^2PROM.

OPERATION

It is necessary to configure the programmer for use with Intel 2716 devices. The adapter is inserted into the 2716 socket on the programmer and the E^2PROM is then placed into the Textool socket on the adapter board.

Reading from the 2816 is done exactly as it would be for a 2716. The socket adapter switch must be in the PROGRAM position.

To write bytes of data to the E^2, it is first necessary to ensure that the bytes are erased (FF hex). Chip erase mode is engaged by placing the socket adapter switch in the ERASE position. Subsequently, a program cycle is initiated by the programmer at any 2816 address with any data. Upon the assertion of the first V_{PP} pulse, the entire 2816 array will be erased. Internal circuitry on the E^2PROM OE line is responsible for determining the chip erase operation. After the entire chip has been erased, it can then be programmed with data by placing the switch in the PROGRAM position. Data is simply programmed with the device at the desired locations.

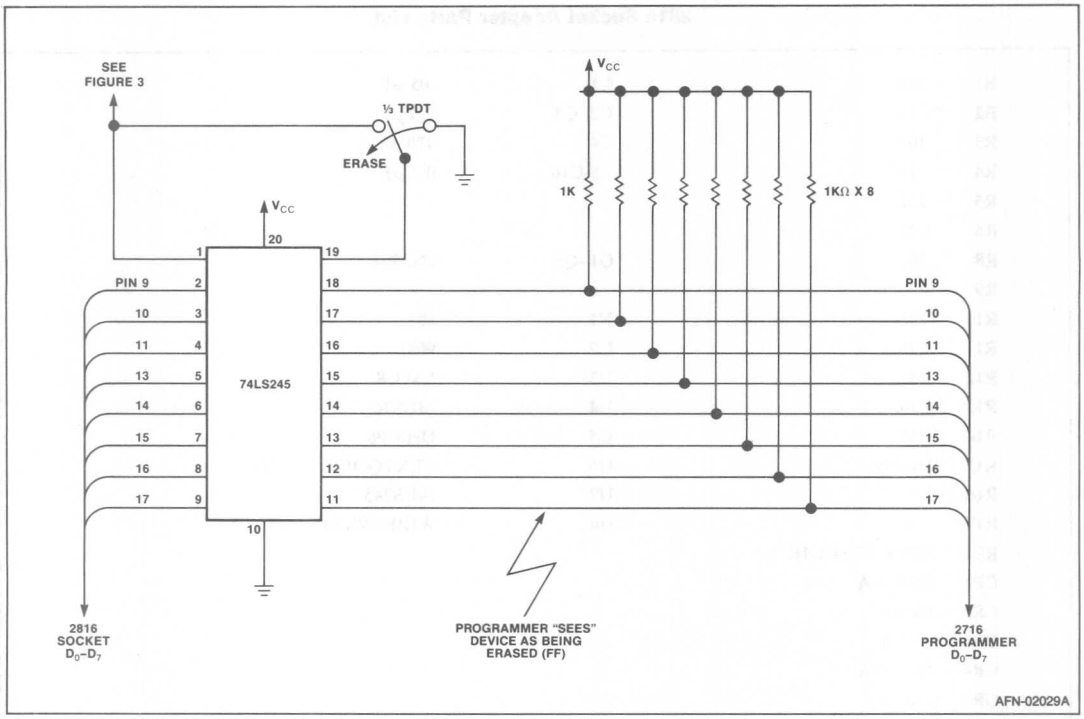


Figure 5. Erase Conditioner

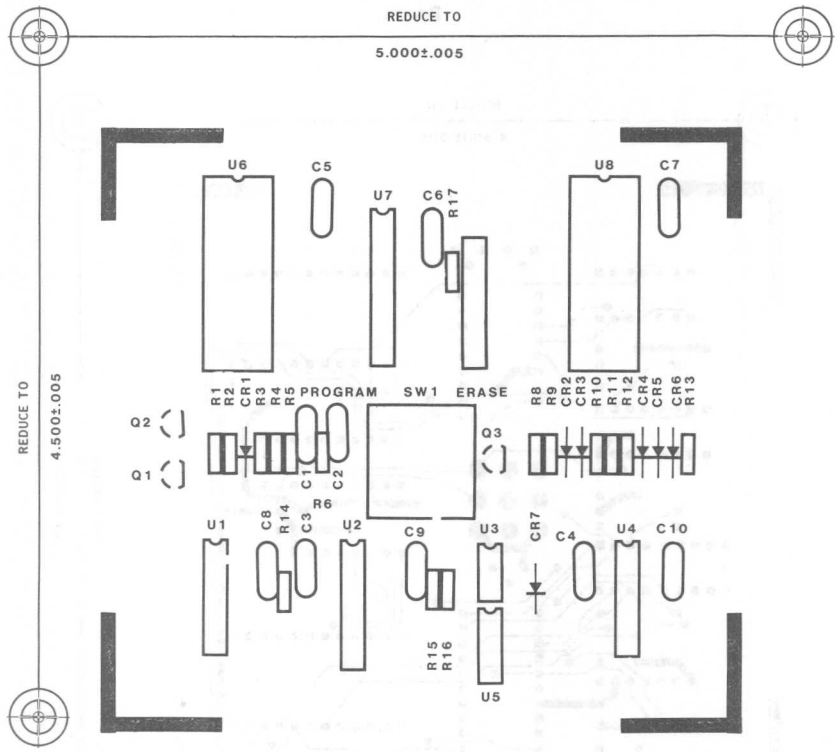
Some programmers will not generate a V_{PP} pulse or else skip over a given address if the data is FF (Pro-Log and DATA I/O are examples). For this reason, some type of data must be "programmed" into the device prior to the chip erasure mode. Since the device will become erased as expected by the user but not the programmer, the 2816 will fail subsequent verifies and an error message will ensue. This is typical of most programmers.

CONCLUSION

E^2PROM is ideally suited in applications requiring that program store or data be revised occasionally. The 2816 socket adapter allows the user to program E^2PROMs on his existing EPROM hardware.

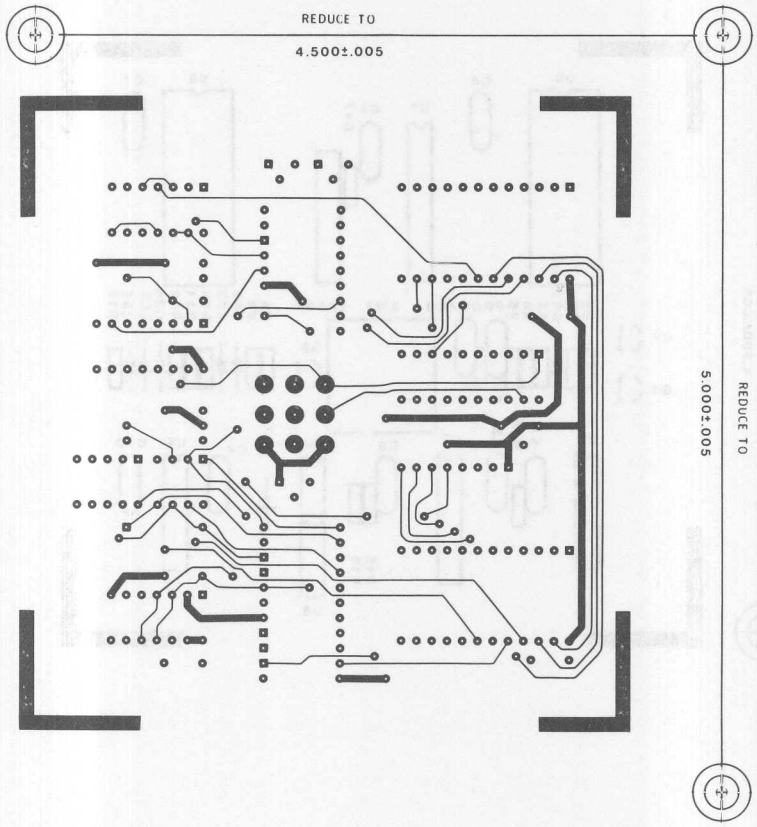
2816 Socket Adapter Parts List

R1	22K	C1	.05 μF
R2	15K	C2, C3	1.0 μF
R3	10K	C4	470 pF
R4	1K	C5-C10	0.1 μF
R5	10K		
R6	1.2K		
R8	10K	Q1–Q3	2N3904
R9	1K		
R10	10K	U1	4011
R11	22K	U2	9602
R12	15K	U3	LM358
R13	4.7K	U4	74LS74
R14	51K	U5	DH0006
R15	1N4748	U6	TEXTOOL SOCKET
R16	2.7	U7	74LS245
R17	1K	U8	WIRE-WRAP PINS
RP1	SIP X 7 764-1-1K		
CR1	1N4739A		
CR2	1N914		
CR3	1N914		
CR4	1N4739A		
CR5	1N914		
CR6	1N914		
CR7	1N914		
CR8	1N914		
S1	TPDT		


REDUCE TO
5.000±.005
REDUCE TO
4.500±.005
U6
C5
U7
C6
R17
U8
C7
R1
R2
CR1
R3
R4
R5
PROGRAM
SW1
ERASE
R8
R9
CR2
CR3
R10
R11
R12
CR4
CR5
CR6
R13
Q2
Q1
C1
C2
R6
Q3
U1
C8
R14
C3
U2
C9
U3
CR7
C4
U4
C10
R15
R16
U5

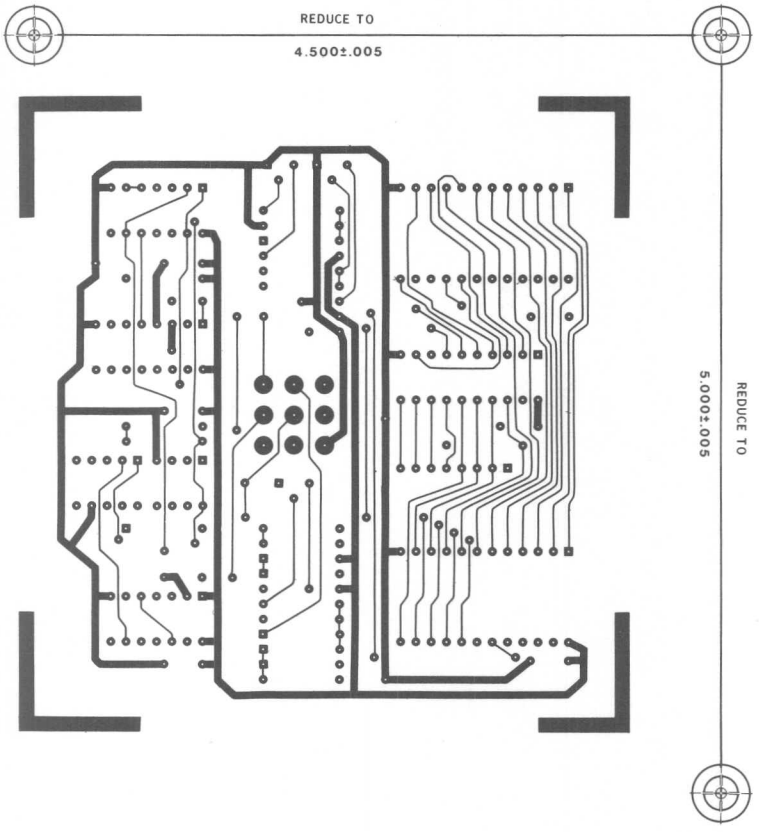
AFN-02029A

BACK



AFN-02029A

FRONT



intel

ARTICLE REPRINT

AR-174

June 1981

EE-PROM Goes to Work Updating Remote Software

Randy Battat and John F. Rizzo
Special Products Division
Applications Engineering

Reprinted from ELECTRONICS, May 19, 1981. Copyright 1981, McGraw-Hill Incorporated. All rights reserved.

AFN-01961A

EE-PROM goes to work updating remote software

Analog signals transmitted over telephone-line data links alter on-site memory for microsystem upgrading and maintenance

by Randy Battat and John F. Rizzo, *Intel Corp., Santa Clara, Calif.*

☐ Microprocessor system software needs frequent revision, which is inconvenient, difficult, and costly. But because it combines the nonvolatility of ROM and the flexibility of random-access memory, an electrically erasable programmable read-only memory at the microprocessor site allows remote software changes to be made through a telephone-line data link, eliminating field service expenses.

As technology progresses, design and service costs are coming to determine—more than component costs—the cost of microprocessor systems (see "The cost of software service"). Intel's 2816 EE-PROM not only solves the service problems, but it also makes existing designs more functional since they need only be updated, not replaced.

Memory requirements

In a remotely controlled EE-PROM, the memory must be nonvolatile, retaining data even when the host system is powered down. Furthermore, with today's high-speed microprocessor systems such as the Intel 8086-2, the Zilog Z8000, and the Motorola MC68000, only fast memory devices can achieve full throughput. For example, a high-performance 8086-2 system with a zero time wait-state operation requires a memory-read access time of 250 nanoseconds.

Also, as software costs rise, high-level languages will often be used to reduce design time. These languages are memory-intensive, requiring high-density memory chips to effectively contain dedicated system programs without sacrificing printed-circuit board space.

Finally, a remote link-addressable EE-PROM must have read-mostly operation. Normally program memory and certain types of data memory are accessed in a read mode. At times, however, it is necessary to reload an entire program (as in the case of a software revision) or to reconfigure portions of data storage (when only certain parameters need to be changed). Then the ability to write into the memory in the circuit is essential.

The 2816 fills all these user requirements. It is nonvolatile, having greater than 20-year data retention. Its access time of 250 ns is compatible with today's high-speed microcomputer systems. The chip is also electrically erasable on a per-byte or per-chip basis—a true read-mostly memory. It offers users 16,384 bits of storage organized as 2048 8-bit bytes.

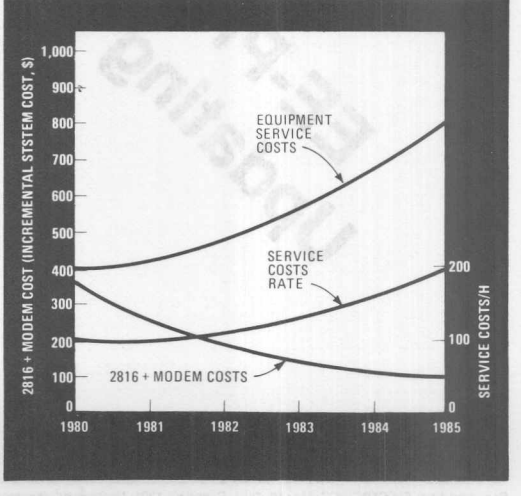
The EE-PROM allows in-circuit erase and write, operations so it can be written into from many information sources. But because it is an excellent medium for storing nonvolatile programs and data, it is particularly suited to downline loading—in this case, in changing memory contents at remote sites via a data link.

The telephone is ideal for transferring information,

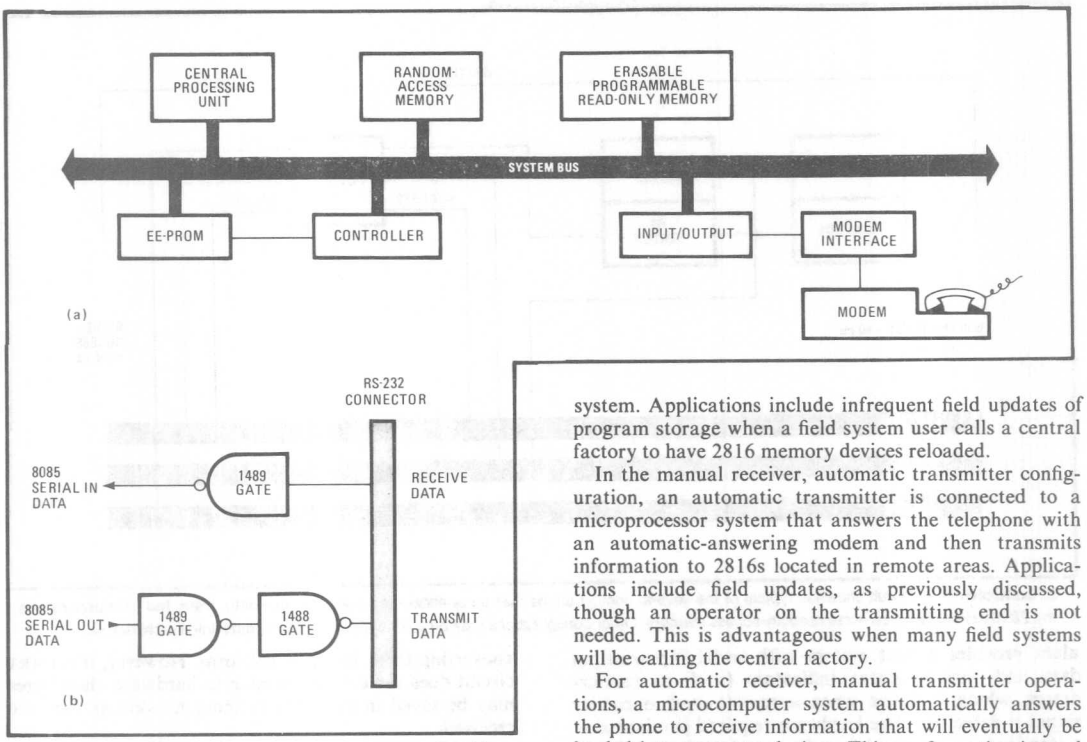
The cost of software service

Servicing a software change in a field application today averages about $100 per hour. By 1985, assuming an inflation rate of 15%, these costs will approach $200/h. A typical microprocessor system (2,000 in the field) requires a service time of 2 h. If each system needs to be updated a minimum of two times during the product's life, the cost is at least $400 per system. That means $800,000 for the total retrofit. If a doubling of the cost of labor in the next five years is assumed, the new retrofitting cost comes to $1.6 million.

By installing a remote-software serial link, software can be updated over telephone lines. Adding a 2816 EE-PROM and a remote link to the system will cost about $50, a mere one sixteenth the 1985 service cost. Today, as seen in the figure, a 40% savings can result.



AFN-01961A



1. Easy downloading. A microprocessor at a remote site with an EE-PROM as a peripheral may have its software changed by means of a telephone data link (a). An acoustically coupled modem is required with an interface (b) that is simple to implement.

since it is readily available and requires no special interface. Using an acoustic coupler, serial binary data is converted into high- and low-frequency tones that are then transmitted over a worldwide link. Modems interface easily with microprocessors, and, in addition, the software overhead for such a downline-loading operation is minimal.

Mixing and matching

Programs downline-loaded into EE-PROMs find many applications in both large and small microcomputer systems. But regardless of size, all configurations require a modem to interface electrical signals from a host processor with the acoustically driven telephone. Automatic modems are usually dedicated to a specific telephone line and are completely operated by a host processor. Manual modems are usually portable, relying on an operator to place a telephone receiver in an acoustic-coupler cradle, thereby closing the communication loop. Both automatic and manual modems can be used in EE-PROM telephone communication systems in four possible configurations.

The first configuration uses a manual receiver, manual transmitter design—a cost-effective solution when telephone transmission is not performed often enough to warrant a dedicated telephone line and microprocessor system. Applications include infrequent field updates of program storage when a field system user calls a central factory to have 2816 memory devices reloaded.

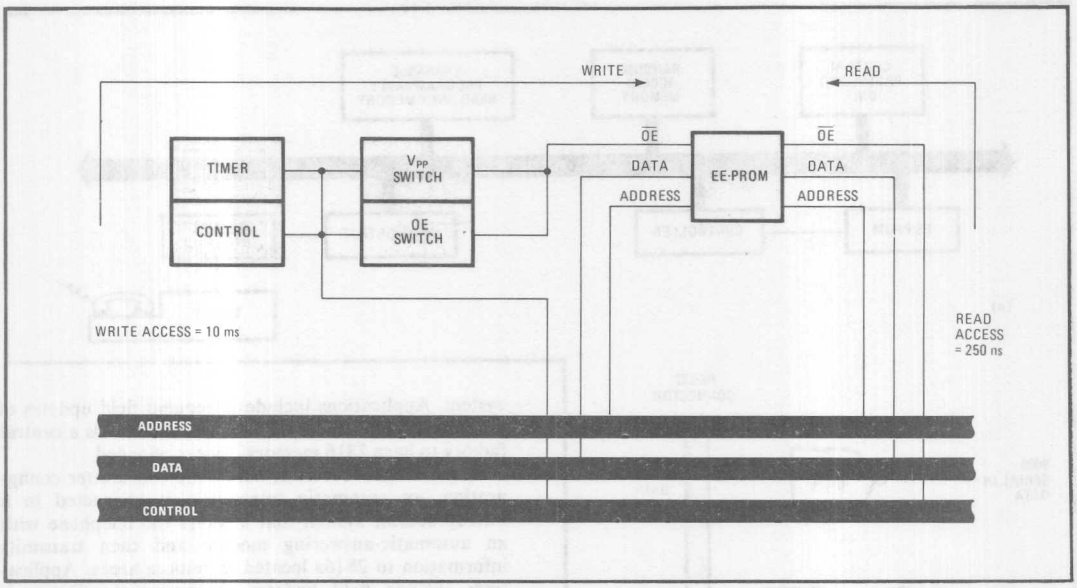
In the manual receiver, automatic transmitter configuration, an automatic transmitter is connected to a microprocessor system that answers the telephone with an automatic-answering modem and then transmits information to 2816s located in remote areas. Applications include field updates, as previously discussed, though an operator on the transmitting end is not needed. This is advantageous when many field systems will be calling the central factory.

For automatic receiver, manual transmitter operations, a microcomputer system automatically answers the phone to receive information that will eventually be loaded into EE-PROM devices. This configuration is used in unattended systems, where, for example, a processor controls remote communications switches or repeaters. If parameters need to be changed, the remote switching processor is telephoned and new parameters transmitted to the EE-PROM in the system. This application exploits the byte-erase feature of the 2816. Only those EE-PROM locations that contain parameters to be changed need be rewritten.

The last configuration, with automatic receiver and automatic transmitter, eliminates the operator. Here an automatic-dialing modem is used. A central computer could be requested to call many remote units to automatically update programs or data in the EE-PROM memory without human intervention.

The hardware elements of an automatic receiver and an automatic transmitter are the same, so describing a system with a manual receiver and an automatic transmitter can help explain all four configurations. Here the human operator on the receiving end initiates transmission by dialing the transmitter and placing a telephone receiver in an acoustic-coupler cradle. The transmitter answers the telephone and transmits data to the receiver. This data is eventually loaded into EE-PROMs.

The significant elements in this configuration are the modem and modem interface, the receiver central processing unit and associated software, and the 2816 and its controller (Fig. 1a). The receiver CPU is connected to a simple modem that converts serial binary data into acoustical tones. The standard Bell 103 modem or equiv-



2. Controller. The circuit shown is typical of the several configurations that an EE-PROM may use to receive data. Here, the controller makes the 2816 resemble a slow-write random-access memory. Other configurations may be used depending on the data link's architecture.

alent provides a host system with serial input/output data and various status indicators (such as carrier-detect, which is active when a remote-modem carrier signal is detected). The hardware required is minimal—a standard modem can be readily purchased. An RS-232 interface is needed to interface 5-volt TTL signals from a CPU I/O port (or serial data line) with the ±12-V RS-232–compatible signals of the modem. Software handles the rest of the downline loading operation.

Figure 1b shows a simple modem interface. The MC1489 converts RS-232 levels into TTL levels, while the MC1488 converts TTL signals into RS-232. Here, serial data I/O lines can be passed directly to a universal asynchronous receiver transmitter (UART) for serial-parallel data conversion. If an 8085 processor is used, the serial I/O lines can be connected to the 8085's serial out-data (SOD) and serial in-data (SID) ports. Another option is to perform the serial-parallel conversion in software. The receiving CPU need only receive data bytes (possibly after a transmitter identification message is received) and program the EE-PROM.

The actual reception of data is simple. The processor first transmits an identifier message sent from the remote transmitter. This latter message may consist of a sequence of binary or ASCII data detailing the location of the transmitter, date and time of transmission, the number of bytes to be transmitted, and the address in EE-PROM of the data to be located. Next, the processor receives a data byte that may be immediately programmed into the 2816 or saved temporarily in RAM. If software performs serial-to-parallel data conversion, data received must be saved in RAM. The 2816 cannot be programmed as each byte is received, since the processor must devote most of its time to receiving data bits and converting them into parallel form. However, if a UART circuit does the data conversion in hardware, data bytes may be saved in EE-PROM memory as soon as they are received.

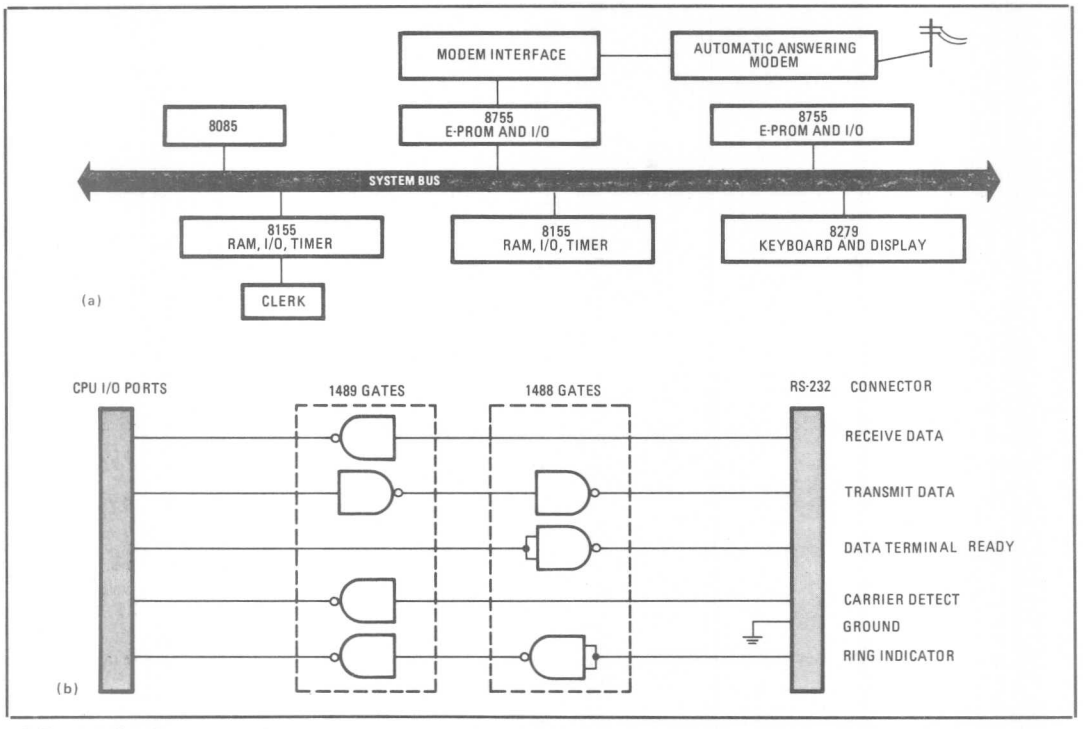
In this process, if data is transmitted at 300 bits per second and if each character consists of 1 start bit, 8 data bits, 1 parity bit, and 1 stop bit, there will be 11 bits per character and a character will be received every 36.7 milliseconds. After every character a 2816 byte must be erased (10 ms) and written (10 ms), leaving 16.7 ms of free time until the next byte is received.

A typical controller

The final consideration in the downline load receiver is a 2816 controller circuit. Figure 2 shows a block diagram of a typical circuit. The read operation for the interface is identical to that for E-PROMs. To read data, chip enable ($\overline{CE}$) and output enable ($\overline{OE}$) are taken low after addresses are set up.

To write into the 2816, the host processor simply writes into memory. The controller circuit pulls the processor ready line low, stalling the CPU and stabilizing addresses and data for the 10-ms write interval while the programming pulse (V_{pp}) is active. With the controller, the 2816 resembles a slow-writing RAM except that it needs byte erase prior to writing.

The transmitter consists of a dedicated microcomputer connected to an automatic-answering modem, which is in turn attached to a telephone line. The transmit computer software loops while waiting for an incoming call. When a call is received the modem is signaled to answer the telephone. Information, in the form of data bytes, is received and transmitted in the same fashion as on the receiving end. Essentially, all the user's transmitter sta-



3. Base station. To transmit software changes to a remote-site EE-PROM, a base station (a) might use an 8085 microprocessor, with a keyboard display to help the user keep track of changes. Standard gates on the modem interface (b) furnish the required control signals.

tion must do is look for a remote-processor identification message, send its own identification message, transmit data serially, and hang up the telephone. Additional features may be implemented, such as a log of all calls received and their origins.

Figure 3a contains a block diagram of a transmitter base station system. An 8085 processor is used, with an additional 512 bytes of RAM and 4-K bytes of E-PROM. A modem interface is shown, in addition to a key pad and display for local-user operation and a real-time clock for logging date and time information.

In this design, the E-PROM memory contains information program storage and transmittal. This is the data that is to be transmitted to remote processor sites. Note that the data transmitting E-PROM could be replaced by an EE-PROM device to allow for frequent changes in transmission data without requiring the physical replacement of the transmit-data store. RAM is used to save logging information, temporary program data, and a character input buffer that stores received characters when a specific message is sought.

The key pad and display module enables a local base-station operator to interrogate the base station and reset date or time, or access a call log. The clock module is used to keep track of current date and time. Such data may be transmitted to remote processors or may be used locally as a part of the information logged pertaining to each call received.

The modem interface for the base station is very similar to the receiver modem circuit. Figure 3b contains a circuit diagram of an automatic-answering modem interface. The circuit provides all signals and takes care of the ring indicator signals. The first is given by the host processor and tells the modem when to answer and hang up the phone. The second is active when the phone is ringing and is used to interrupt the processor.

A real circuit

A base station similar to the one described has been constructed at Intel. It is used to transmit information to remote 2816s for demonstration purposes. In this unit, the software has three operating modes. The first, the interactive mode, is the default, in which the processor displays the time of day while waiting to enter either the dial-in or local-user mode.

The dial-in mode is entered whenever a call is received. The processor answers the line, looks for a remote-processor identification message, and transmits its own identification header, followed by text data to be loaded into EE-PROM memory. The telephone is hung up as soon as transmission is completed and the inactive mode is entered. The local-user mode contains software implemented through the local keypad-display to allow a local user to reset. □

EDITORIAL

A revolution in non-volatile memory

With this first PREVIEW issue of 1981, Intel is happy to announce the 2816–a 2K x 8 electrically erasable PROM. During the past ten years, Intel has developed EPROMs to meet the needs of the most demanding customer systems. The quest for a perfect non-volatile memory has been led by Intel from ROM to PROM to EPROM and now, after intense development to the E²PROM. The E²PROM technology promises to alter dramatically the microprocessor systems of today and offer end-users greatly enhanced flexibility and system cost-effectiveness.

How often have you had to make changes to software in the field? Unfortunately, most of us would answer "too often" to such a question. How many times have you envisioned putting added value and capability into your designs, but did not have a memory solution to the problem? Again, the answer might be "too often."

The 2816 will dramatically change the answers to those questions. In the realm of software updates, the 2816 will be used to contain system software, diagnostics, or patch tables. When a change needs to be made, a central computer can establish a link to the unit over a telephone line without needing to send a person to service the equipment. Figure 1 shows the cost trends for such a system. If a remote line cannot be established, then the service technician only needs to plug a cable into the equipment and make a very rapid and simple change; no physical tampering is necessary.

A good example of one way to use E²PROMs is illustrated by Intel's Color Graphic Terminal, a demonstration unit designed with advance Intel microprocessors, memories and peripherals (see sidebar on page 5). This intelligent CRT presents an eight-minute program of sophisticated graphic displays using eight different colors. An E²PROM controller board in the unit allows terminal reconfigurations to be carried out using keyboard interfaces instead of hardware switches. In addition, communications protocols can be transmitted to the terminal through remote data-link.

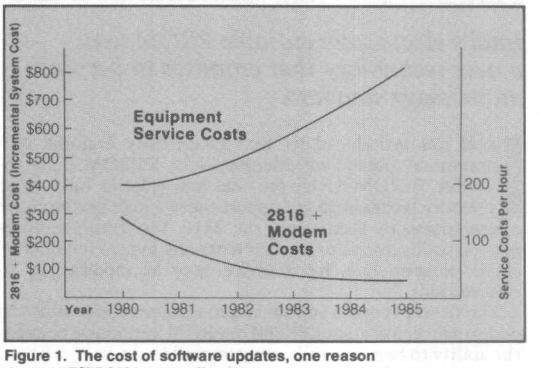


Figure 1. The cost of software updates, one reason the 2816 E²PROM is cost-effective.

Another key application is in industrial and process control equipment. The Intel iSBC 88/40 Measurement and Control Computer (appearing on page 6 of this issue) uses 2816s to store process parameters and control points.

With regard to adding functions and benefits to your systems, only you can understand the doors that the 2816 will open. Intel is committed to the technology of electrically eraseable PROMs and we see it as truly a revolution in non-volatile memory.

John Rizzo
2816 Product Manager
Special Products Division
INTEL CORPORATION

PRODUCT FEATURE

A new generation of memory devices is led by the 2816

Intel's electrically erasable PROM uses a new technology that promises to become an industry standard.

Intel has introduced its first Electrically Erasable Programmable Read-Only Memory—the E²PROM. This 16-kilobit E²PROM device provides non-volatile, fully static memory fast enough to support today's high-performance microprocessors. Designated the 2816, the device requires only 10 milliseconds to erase or write any byte of memory. A single program line, for example, may be modified in 20 milliseconds.

The device is quite similar in pin characteristics and read performance to current EPROM or PROM memories, but adds the ability to be electrically programmed in the field, without being removed from the system. The device can even be reprogrammed remotely, via a radio or telephone link.

This flexibility permits design engineers to create applications that were impossible or too expensive with less flexible program storage devices. Design advantages include reduced system downtime for program changes, faster modifications to stored programs through byte-for-byte replacement capability, and excellent prototyping characteristics.

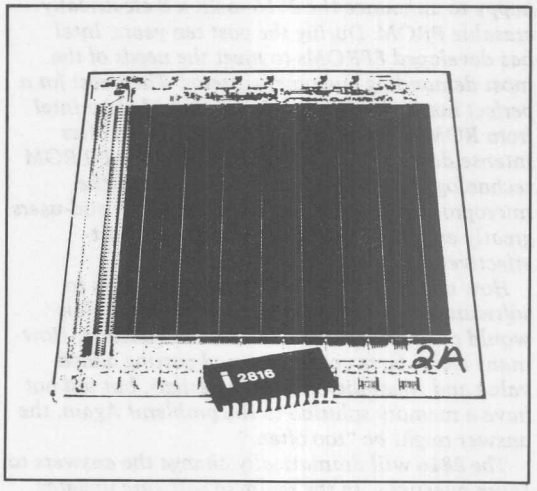
The 2816 requires only the application of a 21-volt pulse for 10 milliseconds to erase or write any byte of memory. The only hardware needed to interface the 2816 to a microprocessor is a programming pulse generator and timer circuit. Because the E²PROM delivers individual byte-erase capability, the end user can make a single-line program edit in 20 milliseconds—as much as 100 times faster than systems requiring bulk-erase and complete rewrite.

The 2816 was a well-planned advancement in erasable program-storage technology. The device is delivered in a 24-pin package that conforms to the new industry-standard pinout for high-density, Byte-Wide memories recently approved by the Joint Electron Device Engineering Council (JEDEC). It is pin-for-pin compatible with devices like the 2732 and 2764 UV EPROMs. By using the 2816 and printed circuit boards with 28-pin sockets, system designers can be assured of future compatibility and interchangeability of microcomputer system memory components of up to 256 kilobits in density.

Performance now

The 2816 E²PROM has both the speed and controllability required for service with today's high-speed microprocessors and microcomputer systems. Maximum access time is 250 nanoseconds, allowing the design of systems without "wait states" in the microprocessor's program.

The 2816 also features two-line control—a system control function essential to large, high-speed microcomputer systems. Two-line control eliminates contention between addresses and data on bus lines. The device has separate output-enable and chip-enable pins, permitting the microprocessor to control exactly when the chip is enabled.



As designers learn to use the flexibility of the E²PROM, the device will move down the production learning curve. By 1985, the E²PROM will be as inexpensive as the UV EPROM; by that time, they will replace the EPROM as the program storage medium in microprocessor-based equipment. In the interim, E²PROMs will be designed into applications where the presently higher cost is offset by the functional value of the device, and the flexibility added to the end user's products.

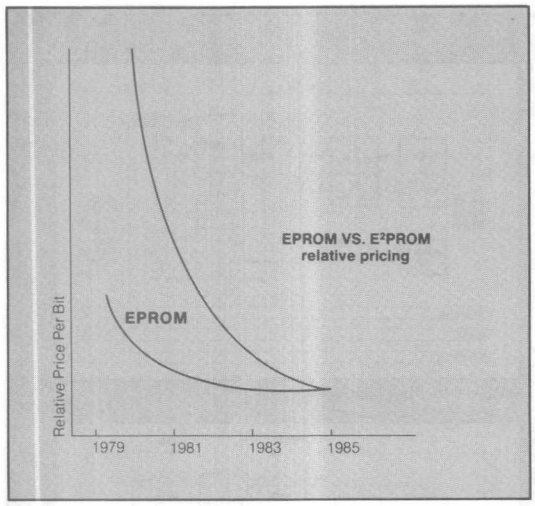
Endless applications

One market segment that will find E²PROMs attractive immediately is industrial process control. In large plants with distributed processing stations under control of a central computer, E²PROMs can improve local process monitoring and control.

In such configurations, the central computer alters the E²PROM contents remotely when a process change occurs, to optimize local processor operation to the new conditions. The E²PROMs can also be used as data storage devices to monitor flow rates, value closures, and like information, freeing the central computer for other duties.

Another obvious application for E²PROMs today is replacements for core memory or fuse-link PROMs in military equipment and commercial aircraft. Here, the in-system flexibility dramatically increases overall efficiency when compared to the time and cost of replacing expensive parts each time a change in flight coordinates or radio frequencies is required.

Point-of-sale terminals are an ideal E²PROM application. The devices function as lookup tables whose contents—prod-



uct pricing, for example—change frequently. The computer can poll and update the E²PROMs after business hours at the retail store, to monitor sales volumes and adjust pricing.

Another immediate application for E²PROMs is in programmable robots like those used in automobile manufacturing or industrial metalworking. Presently, program changes require replacement of the paper or magnetic tape that controls the robot's operation. And E²PROMs offer superior reliability in harsh industrial environments in addition to the in-system programmability that will dramatically reduce retooling charges.

General areas where E²PROM devices will find use include applications where continuous calibration, dynamic reconfiguration, programmable mapping, remote communications reprogramming, CRT terminal configuration, CRT graphics configuration, military replacement, printer control, or remote data logging are required.

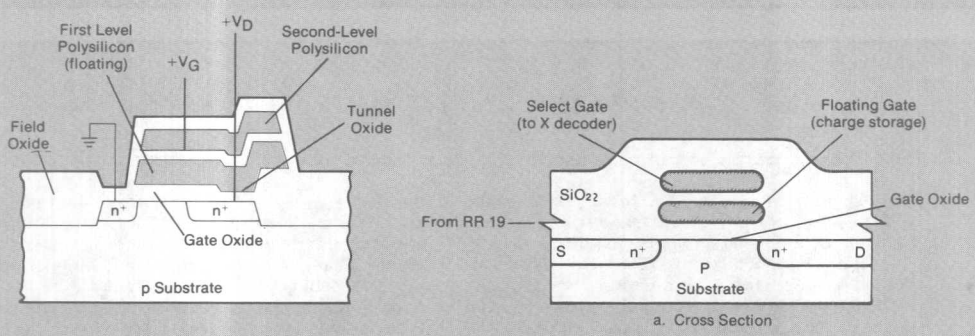
Flotox cell structure, an extension of EPROM

Intel's new flotox (floating-gate tunnel oxide) cell structure employs a mechanism called Fowler-Nordheim tunneling to write and erase data. The 2816 erases and writes by causing electrons to tunnel across a thin (less than 200 Angstrom) layer of silicon dioxide. The 2816's cells hold their charge in the same way as Intel's conventional EPROMs. At 125 degrees Centigrade, the 2816 will retain data for at least 20 years. The device is fully static; refreshing is never required, regardless of read frequency.

Data stored in the 2816 is easy to modify. Byte erase/write or chip erase requires application of a 21-volt pulse for 10 milliseconds. Since it is byte-erasable, a single-byte program change can be made on the 2816 as much as 100 times faster than it can be made on a bulk-erase part. Any of the 2 kilobytes of the 2816 can be erased and rewritten in 20 milliseconds.

Rated for operation from 0–70 degrees Centigrade, the 2816 is a low-power device, dissipating 495 milliwatts from the single +5-volt read supply when active, and 132 mW when in standby mode. The chip goes into standby automatically when not enabled.

Four controller configurations provide the designer a choice



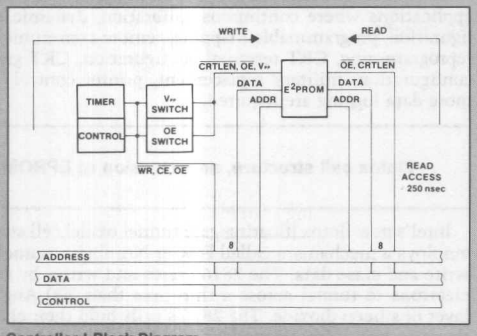
2816 Cell Structure

Fowler-Nordheim tunneling is bilateral. The direction of the electric field determines whether electrons tunnel to or from the floating gate. The floating gate is surrounded by silicon dioxide, giving the 2816 a 10-year data retention.

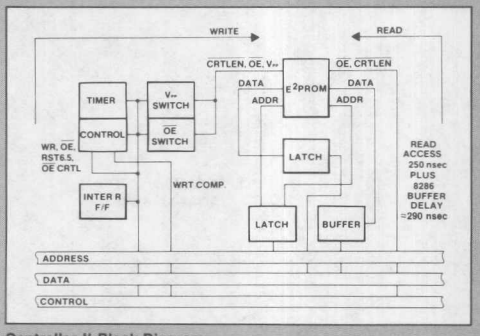
HMOS-E technology is used to produce 650 Angstrom gate oxide and 200 Angstrom tunnel oxide thickness.

EPROM Cell Structure

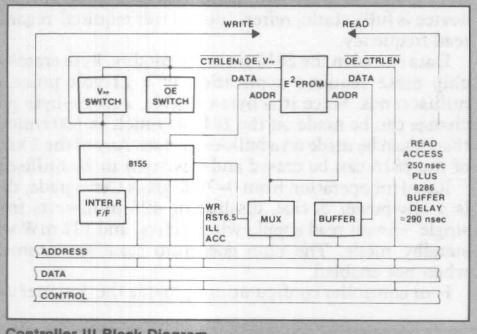
Programming is accomplished by placing electrons on a floating gate through hot electron injection. Erasing is accomplished by exposing the device to UV light for 30 minutes. The charge can't leak off because the floating gate is surrounded by silicon dioxide.



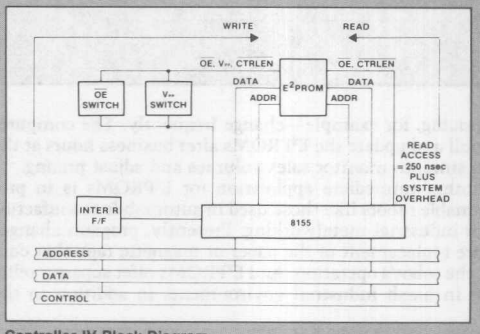
Controller I Block Diagram



Controller II Block Diagram



Controller III Block Diagram



Controller IV Block Diagram

A big step in program storage

The first program storage device was the masked read-only memory, or ROM. Masked ROMs are programmed by the semiconductor manufacturer with instructions supplied by the OEM buyer. Once programmed, they cannot be altered. Each program change, therefore, requires the purchase and manufacture of a new ROM device, which may take several months. ROM devices are inexpensive in large quantities, but require a large initial investment and a large quantity commitment.

Next came the programmable ROM, or PROM. PROMs can be "burned" by the OEM or end user, but can be programmed only once. They are costlier than ROMs on a per-unit basis, but they eliminate dependence on semiconductor manufacturers for programming.

Erasable PROMs, or EPROMs, added considerable flexibility. Like PROMs, EPROMs can be stocked and programmed by the OEM or end user. But they can be programmed many times. This eliminates the waste of PROM devices when program changes are required. EPROMs were originally envisioned as a development tool for designers, who need to change programs frequently while prototyping and debugging systems. Today they are frequently shipped in production equipment due to their potential value to the user, who may wish to make a program change. EPROMs have, in fact, become the most popular program-storage memory.

A drawback to EPROM devices is that they must be removed from the equipment to be reprogrammed. (EPROMs are erased optically, through exposure to ultraviolet light, then rewritten electrically with the new program.)

E^2PROMs provide all the benefits of the popular EPROM, without the drawback of removal for reprogramming. The E^2PROM can be electrically reprogrammed by the OEM or end user, but without removal from the system, and without use of exterior programming devices such as PROM programmers.

	16K E²PROM	16K EPROM
Configuration	2K x 8	2K x 8
Package	24 Pin	24 Pin
Power Supplies		
Read Mode	+5	+5
Erase/Write	+5, +21	+5, +25
Write		
Method	Tunnel Injection	Hot Electron Injection
Time/Word	10 ms	50 ms
Erase		
Method	Tunnel Injection	UV Light
Time/Word	10 ms	—
Time/Chip	10 ms	30 min
Access Time	250 ns	450 ns
Power Dissipation		
Active	500 mW	550 mW
Standby	100 mW	100 mW
Data Retention	>10 Years	>10 Years
Refresh Requirement	None	None

of trade-offs between microprocessor burden and controller device count.

Controller I requires only five IC devices, and reads at full system speed. Writes, however, stall the microprocessor for 10 ms. The controller is for use when processing and control speed are not important.

Controller II requires eight IC devices, employing software-driven read/write to improve writing speed. Controller III requires twelve devices, and provides close to real-time writing capabilities. Controller IV requires thirteen devices, and reduces the software overhead for true real-time high-speed processing systems.

Color Graphic Terminal displays Intel's latest technology

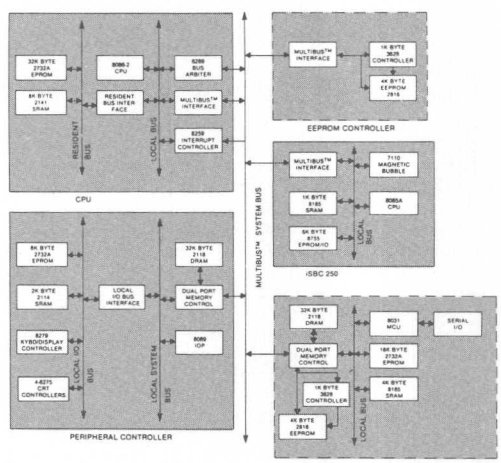
Intel's Color Graphic Terminal is a demonstration unit configured with high-speed, high-performance Intel components. The entire system architecture is based on the MULTIBUS bus system.

The processor module features a 16-bit iAPX 86 microprocessor operating at its full specification of 8 MHz, and a 200 nanosecond 2732A EPROM. Which allows the microprocessor to run at top speed without wait states. This combination makes our system yield the highest throughput in the 16-bit microprocessor category.

The peripheral module uses an iAPX 86/11 I/O Processor. The IOP is designed to process I/O functions with the benefit of intelligent DMA (scan/match, translate, variable terminate conditions), modular programming in a full megabyte of memory address space, and a set of optimized I/O instructions that are unavailable in conventional CPUs. DMA rates of up to 1.25 megabytes/second are possible with the iAPX 86/11. This capability makes it very useful to CRT terminal applications where fast DMA is required for screen refreshing.

With an E²PROM controller module, terminal reconfigurations can be carried out using the keyboard interfaces instead of hardware switches. Using the communications module, program or data can be stored in E²PROM from a remote location without worrying about volatility. A magnetic bubble memory board is used to store data where real-time access is not important. It provides a convenient and inexpensive medium for storing non-volatile data.

The Color Graphic Terminal is available for demonstration to customers. Call your local Intel sales office to set up an appointment to see it.



The Color Graphic Terminal achieves a 250 ns minimum access time using an 8 MHz iAPX 86 microprocessor and high-speed memory components.